UNIVERSITY OF KWAZULU-NATAL

Factors that influence proficiency in the learning of computer programming by Information Technology students

By

Lerushka Ranjeeth

215050307

A dissertation submitted in fulfilment of the requirements for the degree of Master of Commerce

School of Management, IT and Governance

College of Law and Management Studies

Supervisor: Prof. Indira Padayachee

2022

DECLARATION

I, Lerushka Ranjeeth declare that

(i) The research reported in this dissertation/thesis, except where otherwise indicated, is my original research.

(ii) This dissertation/thesis has not been submitted for any degree or examination at any other university.

(iii) This dissertation/thesis does not contain other persons' data, pictures, graphs or other information unless specifically acknowledged as being sourced from other persons.

(iv) This dissertation/thesis does not contain other persons' writing unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:

a) their words have been re-written, but the general information attributed to them has been referenced;

b) where their exact words have been used, their writing has been placed inside quotation marks and referenced.

(v) Where I have reproduced a publication of which I am author, co-author or editor, I have indicated in detail which part of the publication was actually written by myself alone and have fully referenced such publications.

(vi) This dissertation/thesis does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the dissertation/thesis and in the References sections.

Signed:

Date: 03/03/2023

ACKNOWLEDGEMENT

I would like to thank my supervisor Professor Indira Padayachee whose never ending encouragement, wisdom, guidance, and support helped me to produce this research.

To my family and friends, I would like to thank you for the consistent support throughout this journey. Your encouragement and helpfulness have paved the way to the completion of this research project.

ABSTRACT

The main objective of the study was to ascertain the factors that influence the acquisition of computer programming skills by students who are enrolled for an Information Technology (IT) degree at a tertiary education institution. The study is driven by a societal need to empower as many individuals as possible with computer programming skills. The study is very relevant to the South African context in the light of the decision taken by the education department to establish computer programming as a niche skill for South African citizens. The learning of computer programming is however, not that straight forward and requires an intensive cognitive effort to ensure that students obtain a high degree of skill and expertise in computer programming. The study has been conducted at the University of KwaZulu-Natal (UKZN) where the Discipline of IT has been challenged by students' performances in computer programming assessment. While there are "pockets" of excellence, there are numerous instances where students have performed poorly in computer programming assessment. The case of UKZN presents an ideal opportunity to study this phenomenon because it provides a diversified student population with regards to degree enrolment as well as gender and location. From a teaching and learning perspective, this knowledge will be pivotal for the IT academic department at UKZN as well as the general domain of teaching and learning of computer programming.

The study adopted a quantitative approach and was guided by a conceptual framework. The study used a questionnaire that contained an open-ended question that enriched the analysis and discussion. The study's main objective was to ascertain factors that will predict computer programming performance was achieved. The main factors that were identified as significant predictors of computer programming performance were problem solving ability and self- efficacy. A concomitant outcome from the study was the analysis of validity of the study's conceptual model which was subjected to multiple regression and path analysis. The path analysis exercise resulted in the generation of a conceptual model that had a better fit to the study's data than the a priori conceptual model. The study also discovered trends of computer programming strengths and weaknesses at UKZN and it is envisaged that this knowledge will contribute to enhance computer programming pedagogy and student performance in assessment tasks.

TABLE OF CONTENTS

List of Figures

List of Tables

LIST OF ABBREVIATIONS

4IR	Fourth Industrial Revolution
AES	Agriculture, Engineering and Science
ANOVA	Analysis Of Variance
CFA	Confirmatory Factor Analysis
CFI	Comparative Fit Index
FTF	Face to Face
IT	Information Technology
KS	Kolmogorov-Smirnov
LMS	Learning Management System
LS	Learning Styles
MCQ	Multiple Choice Question
OL	Online Learning
OO	Object Orientated/ Object Orientation
PPMC	Pearson Product Moment Co-efficient
RMSEA	Root Mean Square Error of Approximation

POPI	Protection of Personal Information
SE	Self-efficacy
SPSS	Statistical Package for the Social Sciences
SW	Shapiro-Wilk
TLI	Tucker Lewis Index
UKZN	University of KwaZulu-Natal

CHAPTER ONE – An Introduction to the Study

1.1 Computer Programming in Society

Computer programming skill has acquired an elevated status in society in general. The reason for this phenomenon is the ubiquitous presence of computers in society coupled with the notion of a 4th Industrial Revolution (4IR) where computing power in the form of robotics, artificial intelligence and data analytics will provide the infrastructure for economic well-being and business competitiveness. Ristić et al. (2016) stated that the reason programming is taught to students is so that eventually they will be able to secure high paying jobs. Kori, Pedaste, Niitsoo, et al. (2015) found that the common reasons why students chose to study Information Technology at a higher education institution was:

- Inherent interest in the information technology field or working in this field
- The information technology knowledge is necessary for a particular job that they had interest in pursuing
- Financial stability and earning a suitable salary
- To have a better chance in the working market and the possibility of many job opportunities
- Job stability as the field of technology is promising in today's job market and will remain promising in the future

A multitude of these factors have propelled an increasing number of students to register for technology-oriented courses that provide a substantive exposure to computer programming content. In addition to these factors there has also been a significant push for students to learn how to program. Vee (2013) argued that since the 90's computer programming professionals have stressed the need for future generations to equip themselves with a computer programming skillset. The importance of learning at least a single computer programming language has been equated to the acquisition of basic skills in numeracy and literacy.

The rise in student enrolments for technology related courses has unfortunately been paralleled by an increase in failure and poor performance in these courses. The main contributor to this phenomenon is poor performance in computer programming assessment (Govender, 2021). According to Butler and Morgan (2007), 1st year information technology (IT) students face an assortment of challenges. The challenge of adjusting to a completely new environment and learning style which is due to the transition from standard schooling to the tertiary education environment and also the challenge of learning a completely new form of language, which is computer programming, that many students have never been exposed to before. Students have consistently performed poorly in programming assessments at tertiary level and university courses with programming involved in the curricula have also experienced significantly high dropout rates (Kori, Pedaste, Tõnisson, et al., 2015; Luxton-Reilly et al., 2018). According to Kinnunen (2009) and Medeiros et al. (2018) a large number of institutions record high rates of failure in introductory programming courses.

This phenomenon calls for an investigation into what precisely has an influence on students' performance in introductory computer programming courses and what factors could lead to these students acquiring the requisite skill-set to become a competent computer programmer. This study will attempt to investigate and outline the main factors that have a significant influence on students' programming skills by conducting a quantitative analysis in the form of surveys to an audience of students enrolled in an introductory programming course in the discipline of the Information Technology at a tertiary educational institution.

1.2 Background of the Study

The industrial world has been inundated with a demand for computing based skills and the focus area has been in the domain of computer programming and software development (Dirzyte et al., 2021; Konecki & Petrlic, 2014). According to Abdunabi et al. (2019), the specific demand is for skills that translate to job roles such as business analytics, data analytics, project management, software engineering, software development and testing, systems analysis and design, database and network administrators. The demand in these skills is expected to escalate by an amount of 13% in the period from 2016 to 2026 (Abdunabi, Hbaci, & Ku, 2019). Central to all of these job portfolios is either a deep or conceptual understanding of computer programming.

Computer programming has always been regarded as a challenging field of study, currently and historically (Alturki, 2016; Cheah & Leong, 2019; Hassinen & Mäyrä, 2006). Failure and dropout rates for introductory programming modules have always been a topic of concern within the academic community. Bergin and Reilly (2005) produced seminal papers based on empirical evidence, attesting to the cognitive challenges that students have as they learn how to write computer programming code and this compromises their chances of achieving a successful outcome. This phenomenon is philosophically explained by Guzdial (2010, p. 1) by suggesting that computer programming is "… *innately one of the most complex cognitive tasks that humans have ever created*". As a result of the claim that computer programming is a challenging discipline to master, there have been many studies that have attempted to identify factors that contribute to the acquisition of competency in computer programming. Kazimoglu et al. (2012) have observed that students who are novice programmers tend to view the activity as purely technical. Due to this, students go about the task of programming in a superficial way without acquiring a deeper understanding of the intricacy required to develop a successful programming solution to a problem. This factor deals with the problem-solving ability of the students as well as their mental model abstraction of the problem domain. The challenge of engaging in abstract thinking and the ability to create an accurate mental model of the problem domain will have a compromising influence on students' aspirations to be successful in computer programming. However, the cognitive dimension in computer programming is far more complex than just an ability to be a good problem solver. There is also the influence of psychological factors such as self-efficacy and a general attitude towards the task of computer programming. Lee et al. (2017) conducted a study on students' performance in an informatics test that contained computer programming tasks and discovered a significant positive correlation between perception and attitude towards computer programming and their performance in the test. Kong et al. (2018) suggested that obtaining mastery in computer programming entailed the acquisition of competency in being able to establish meaning/comprehension of a problem situation, understanding the impact of the problem and the solution, being creative in solution development and possessing computer programming self-efficacy. Similarly, factors such as learning styles and motivation levels, previous experience, self-efficacy and problem-solving ability have received attention from previous studies (e.g. Corney et al., 2010; Hoda & Andreae, 2014; Nikula et al., 2011; Robins, 2010). However, there has been a dearth of literature regarding an amalgamation of these factors or the predictability of these factors in enabling the academics to have a sense of anticipation of how students will perform in computer programming assessments.

1.3 The Study's Research Problem

What factors influence the academic performance of computer programming by Information Systems and Technology (IS&T) students at the University of KwaZulu-Natal (UKZN)?

The study's research problems are contextualised by a conceptual framework that enabled the researcher to "unpack" the essence of the main problem statement underpinning the current study (see Chapter 2, Figure 2.1). The conceptual framework was guided by the study's literature review and was used to ensure that the path of the study maintained its focus on the study's research questions. The study's sub-problems were guided by the conceptual framework.

Research Questions

1. How does ***problem-solving ability*** influence IS&T students' performance in computer programming at UKZN?
2. What is the influence of ***self-efficacy*** on IS&T students' performance in computer programming at UKZN?
3. What is the influence of ***learning approaches*** on IS&T students' performance in computer programming at UKZN?
4. How does ***intrinsic motivation*** influence IS&T students' performance in computer programming at UKZN?
5. How does ***extrinsic motivation*** influence IS&T students' performance in computer programming at the University of KwaZulu-Natal?
6. How can computer programming performance be improved by IS&T students at UKZN?

1.4 The Research Objectives

The study's primary objective was to discover a core set of factors that will predict the performance of students in computer programming assessment at tertiary education level. The empirical phase of the study was confined to the cohort of IS&T students at UKZN due to operational convenience. The study adopted an exploratory stance that provided the researcher with the latitude of engaging with the literature in an open and unrestrained manner. The sub-objectives identified for the research were to:

1. Determine the role of ***problem-solving ability*** on students' proficiency in computer programming.
2. Determine the role played by ***self-efficacy*** on students' proficiency in computer programming.
3. Determine the influence of ***learning approaches (deep and surface)*** on students' proficiency in computer programming.
4. Ascertain how intrinsic and extrinsic ***motivation*** influences students' proficiency in computer programming.
5. Identify ***general factors*** that will contribute towards the acquisition of proficiency in computer programming.

The study's set of research questions and objectives was contextualised by the conceptual model (illustrated in Figure 2.1 on page 30) adopted for the study.

1.5 Importance/Significance of the Study

The demand for computer programming expertise has been elevated by the emergence of the 4th Industrial Revolution (4IR) that has compelled society to become intelligent users of technology. Technological intelligence is embedded into 4IR systems such as robotics, data analytics and artificial intelligence (AI) by virtue of computer programming logic. Sophisticated use of 4IR technology will only be possible if the user of such technology has comprehensive knowledge of computer programming logic. It is within this context that the directive from the economic sector is to ensure that graduates have a sophisticated understanding of computer programming, thereby enhancing their employability and ensuring that they add value to the sustained imperative to embrace 4IR technologies. Currently, academic studies have not been conclusive or convergent in their contributions towards ensuring that there is a core set of factors that need to be given cognisance, for students to acquire proficiency in computer programming.

The current study attempted to address this impasse by adopting a conceptual framework that has been guided by previous academic literature on the topic. The main outcome from the study was to use evidence provided in the empirical phase of the study to determine the validity of the conceptual framework, and provide the researcher with evidence to provide answers to the study's cohort of research questions as well as to contribute to the academic discourse on the proficiency of computer programming. In addition, the outcome from this study will be used to inform the design and the pedagogical approach adopted for computer programming courses in the IS&T academic curriculum currently being implemented at the UKZN.

1.6 Justification for the Study

According to Tan et al. (2017) many students at tertiary education institutions tend to experience difficulty when it comes to mastering content from technically oriented subjects such as computer science. Butler and Morgan (2007) did identify poor performance in computer programming assessment as a serious problem and it has been a prominent issue since the 90's with many students indicating that computer programming was not all that exciting to learn but it was quite a cognitive burden to engage with and ensure good results. Sunday et al. (2020) found that at a higher institution of learning, 67% of students had failed a module titled, "Introduction to Computer Programming" due to a lack of comprehension of basic computer programming logic. It is evident that computer programming has been known as a difficult subject in the past, however it is also evident that the trend has continued to the present day. It has become crucial for further research to ascertain why students are struggling with this subject in particular. By gathering more knowledge, the reasons for the poor programming performance can be understood and analysed to draw conclusions. This knowledge/understanding will enable educational institutions to implement measures to reduce poor performance in computer programming and encourage more students to pursue computer programming as a career or a course of study. This approach would also have a significant benefit on developing countries as this would empower technical professionals in the country to drive technology focused solutions and thus improve their economy. According to Selamat et al. (2017) there will be a future demand for technical skills such as programming due to the requirements of the 4IR generation. Butler-Adam (2018) opines that in South Africa there will be a growing need for skilled professionals in the technology field within the coming years and an emphasis on learning problem solving and technology related subject matter will assume core focus. The exponential changes to societal behaviour due to the advent of sophisticated technological systems has resulted in a large portion of low-level jobs becoming automated. The prediction made is that the current decline in the need for low- level skills will be replaced with a heavy surge in jobs related to the technology field. The implication in this trend is that there has to be a huge focus on getting a core mass of expertise in fields such as computer programming to ensure a nation's survival in the age of 4IR. There is a corresponding need for educational institutions to place a heavy focus on computer programming pedagogy so that there is ample supply of computer programming skill.

Currently the number and intensity of research efforts on this topic is severely lacking because the myriad of factors that influence the learning and academic performance by students in computer programming, provides a fertile area for research. The current study contributes in this regard by incorporating these factors into a conceptual model that will be subjected to empirical validation.

Durak et al. (2019) stated that programming and learning how to use various applications are becoming increasingly crucial for students to learn both in primary and high school due to programming becoming a critical skill to have in the 21st century. Sarıtepeci et al. (2017) found through a study conducted with students that a group of students who had received training in computer programming fundamentals had also developed greater conceptual thinking ability and out-performed students who did not receive programming training. Durak also points out that critical thinking and problem-solving abilities have become crucial 21st century skills and teaching students programming improves their critical thinking abilities.

The need for computer programming expertise is confirmed in reports by the Australian Government that indicated there is a shortage of skills in the domain of software development. This skill was identified as the one of the three most "in demand" occupations within the professional, scientific and technical services industry in Australia (Australian Government Department of Jobs and Small Business, 2018). This trend has been confirmed in the United States of America (USA) where it is reported that employment for software developers is expected to increase by 24% from 2016 to 2022 (Kanaparan, Cullen, & Mason, 2019).

Coupled with this demand for computer programming expertise, failure rates in academic courses related to computer programming have been reported to be between 30% and 50% (Kanaparan et al., 2019; Quille & Bergin, 2016; Watson et al., 2014) in introductory programming modules.

As Kong (2017) points out, computer programming will be considered to be an indispensable skill in the digital era. It is within this backdrop that the current study had been undertaken. While studies pertaining to pedagogical interventions regarding computer programming have been undertaken in the past, the current study drew from this knowledge to uncover knowledge of how computer programming students engage with the task of mastering computer programming skill in the current era. This knowledge will be crucial in helping educators to plan computer programming courses so that students are optimally placed to receive instruction that is planned according to an insight that is provided by the current study.

1.7 The Study's Limitations

The study may be deficient in external validity because the study is limited to IS&T students from 2nd year up to Master's level at UKZN. While the study examined the phenomenon of how students learn computer programming, this information is highly discipline-specific. The results of the study may not be appropriate in other instances. Another possible limitation in this study is that students may not know how to judge their own computer programming skills very well thereby compromising the integrity of the results with regards to self-efficacy, learning approaches and intrinsic and extrinsic motivation (major constructs used in this study to derive the conceptual framework).

CHAPTER TWO – The Literature Review

2.1 An Introduction to the Literature Review

The literature review section is used to contextualise the study by making a reference to previous studies on this topic as well as reference to key concepts that underline the study.

2.2 Difficulties in learning computer programming

It has been established by Garner that academic performance in computer programming requires a significant cognitive effort from students. However, there are many factors that contribute to this cognitive load and an understanding of these factors is pivotal to ensuring that the failure rates for computer programming assessment is brought under control. The factors that influence the proficiency of computer programming are vast and diverse and range from demographic variables such as gender and previous experience, to psychological variables such as intrinsic and extrinsic motivation to learn computer programming. This constellation of factors forms the basis for the discussion that follows.

2.3 Self-efficacy in computer programming

Self-efficacy (SE) is described as a person's evaluation of their own abilities and skills that they possess and whether or not their competencies may be used to deliver meaningful results that may have a positive effect on their community as a whole (Bandura & Wessels, 1994). Based on this definition, SE is a reference to an individual's confidence in their ability to produce a desirable outcome. SE is a mental characteristic that plays an important role in many aspects of a person's life. There is not a great deal of understanding when it comes to why some students seem to delight in and excel at computer programming while others find it an uninteresting struggle (Ramalingam et al., 2004). According to Govender and Basak (2015), there are numerous factors that affect a person's ability to learn something new, but it is widely thought that attitude and SE towards the subject matter are some of the most important factors in determining one's success in the particular field (D. W. Govender & Basak, 2015). According to Fang (2012), students who experience excess amounts of difficulty in programming may have low levels of self-efficacy which also reduces their motivation towards the subject. Tan et al. (2009) found that throughout the initial stages of learning programming the difficulties met by students, directly shaped their perceptions of programming as a whole. If students were faced with difficulty early on in their experience with programming, they were more likely to adopt an overall view of computer programming as being inherently difficult. This means that students who have had a negative experience with programming earlier on will usually misjudge programming as being difficult and will hold on to that perception usually until the end of their degrees. This inaccurate negative perception of programming can lead to students having low enthusiasm and lower levels of self-efficacy towards the subject. Students who fall into this cycle will unconsciously reject the need to acquire mastery of computer programming thereby acquiring a low level of self-efficacy towards computer programming oriented challenges. According to Askar and Davenport (2009), SE has a profound influence on the activity of learning a new skill and severely impacts the invocation of that skill for problem solving purposes. Askar and Davenport (2009) argued that while an individual may possess the required knowledge and skill to accomplish a task but is lacking in self-belief and motivation, then these impediments will compromise the chances of success.

Self-efficacy and Coding Ability

Wiggins et al. (2017) studied the role that SE played on the quality of the coding that students produced when solving a problem using the Java programming language. Two of the significant outcomes from this study were that male students generally had a higher level of SE towards computer programming in comparison to female students. On an average, students with a higher SE in computer programming tend to produce higher quality code and achieve better performance in computer programming. Wiggins et al. (2017) do however, concede that the limitations of the study such as coding in a specific language and a small sample size do not make the results generalisable. A further analysis of the influence of SE and possibly gender on the attainment of good performance in computer programming is required. These results are corroborated in a similar empirical study by Lishinski et al. (2016) that was conducted on students. The study was based on their academic performance in a computer programming assessment as well as a computer programming project. The main outcome from this study was that SE was the most significant determining factor of computer programming performance. The preceding outcome applies for both, a summative examination-based setting or a formative computer programming project-based setting. Another significant outcome from this study was that female students who had low SE were less likely to recover from this situation thereby leading to a lack of meaningful engagement with computer programming tasks. However, male students who had low SE in computer programming tend to make more of an effort to improve on their computer programming skills consequently also increasing their levels of self-efficacy towards computer programming. Female students were more prone to “internalising” their perceived lack of SE towards computer programming than male students. From a teaching and learning perspective, this is an important observation because the implication here is that introductory computer programming course content plays a pivotal role in ensuring competency and high levels of SE in computer programming, especially for female students.

The role played by SE in enhancing computer programming skills is also confirmed in I. Govender et al. (2014) where a strong link is established between SE in problem solving and SE in computer programming. In this study it was suggested that students who have confidence in their problem-solving ability tend to perform better in computer programming tasks.

A Psychological Perspective

From a psychological perspective, Bresó et al. (2011) explain that students who had lower SE were more likely to have negative beliefs towards themselves and their ability to learn and be successful. These beliefs manifest as stress and anxiety towards their university work and courses. This inferiority complex may lead them to have a negative attitude towards certain subject matter that requires more time and effort to understand. The stress and anxiety around certain subjects may lead students to underperform in these subjects. Bong (2001) endorses the previous opinions by further suggesting that students with a low self-efficacy will be hesitant about setting challenging expectations for themselves in an academic context and will not try to go above and beyond but rather will only do what is compulsory of them. From a computer programming perspective, “doing only that which is required” would ultimately lead to failure because of the complexities involved in mastering computer programming.

Students will only be able to achieve excellence in computer programming by nurturing/acquiring a passion for computer programming. Having a passion for a specific subject matter is especially relevant to the information technology (IT) and computer programming fields because the level of self-efficacy acquired by virtue of students’ engagement with their studies will prevail and impact on their performance as professionals in the working world. Students who have gained confidence in themselves during the course of their university degree will surely outperform the ones who lack self- efficacy and enthusiasm in the workplace.

According to Psycharis and Kallia (2017) educators should try to use pedagogical strategy that can improve students' SE in traditionally difficult subjects that involve problem solving abilities such as mathematics and programming. Psycharis and Kallia also suggest that students' themselves should find methods of boosting their self-efficacy in these subjects in order to develop their skills in these areas to their full potential. Durak et al. (2019) reported that female secondary school students displayed greater computational thinking and programming skills as compared to male students. The author stated that the reason for this difference could be because of the change in perception of gender stereotypes in recent times which has caused female students to improve in their self-efficacy towards mathematical and problem based subject material. According to Wiedenbeck et al. (2007) students who have high SE in programming develop a greater interest in computers which then leads to better programming performance. Kanaparan et al. (2019) conducted a study of 433 programming students where a significant correlation between "programming self-efficacy and emotional engagement" was established for students registered in an introductory programming (IP) module. Kanaparan similarly found a strong connection in the study between students programming SE and their sense of satisfaction and interest in the module which also had a link to students' performance in the module. Lastly, they also found that gratification had an influence on students' interest in programming. Gratification in programming was described in the research as the immediate positive feeling experienced when a student gets a program to run successfully without errors. Typically, when a student has executed and debugged a program that they wrote from end to end, and the result is a fully working piece of software, students report a unique feeling of pride and accomplishment that they reportedly seldom feel when completing projects for other classes. This feeling of gratification has a positive impact on students' interest in programming as they tend to associate the subject with those positive feelings after a few instances of carrying out a program successfully. A study involving 83 secondary school students conducted by Kallia and Sentance (2019) established that those students who did not understand the functions of some core programming statements rate lower in self-efficacy as compared to students who understood these statements well. Presently, educators overlook the importance of their students' belief in themselves and their programming abilities (Kong, 2017). Programming and writing code can be especially demoralising for students due to the frequent errors experienced and the struggle of inserting the correct line of code to execute a program successfully. This is why in programming courses teachers should pay attention to their students' self-efficacy rates/levels because self-efficacy theory states that the more self-efficacy a person has the more resilient to challenges and obstacles they become (Kong, 2017).

A study from Durak et al. (2019) that contained 55 programming students found that female students have better computational thinking abilities as well as greater programming SE than male students. Female students were also shown to be superior at problem solving when compared to the male students. This may be related to one of the study's other findings, which creates a linkage between SE, problem-solving ability and students' preconceived notions about programming itself. This finding suggests that students who enter the field of programming with the preconceived notion that programming is a difficult subject or that programming is reserved only for the intellectually gifted individuals in society may be subconsciously lowering their level of self-efficacy towards the subject. Another factor that may play a role in this is the stigma or association with programmers as people who are socially isolated and develop into hermits who spend most of their time on their computers. This representation of programmers to the general public through media may be adding to the lack of self-efficacy that students feel when they think about programming as a concept. As the majority of young people cannot relate to this stereotyped view of a programmer and this may lead to them believing that they are simply not meant to become good programmers themselves. This view of programming before even beginning a programming class may be the preconceived notions that are leading to lowered self-efficacy in students. Gorson and O'Rourke (2020) conducted a study with 214 computer science students at 3 different universities. The study was aimed at assessing students' self-assessments of themselves when encountering different programming practices such as getting a syntax error or planning. The results of the questionnaire found that some students negatively assess themselves in each scenario of programming which leads to a further negative view of their programming abilities. The study also found that the more frequently a student negatively assesses themselves when performing programming tasks, the lower their self- efficacy tends to be. The study also looked at the students' mental imagery of the competence required to be a professional programmer and found that students who believed that they could not acquire this level of competence had low levels of SE resulting in poor performance in computer programming assessment.

2.4 Previous Experience

Students who are novice programmers may find the task of learning to write computer programming code filled with challenges and difficulties that they would not have anticipated. According to Kori et al. (2016), students who have had previous experience with learning programming, through high school classes or their own independent efforts, perform better in programming courses in university. The reason for this could be because of the programming environment itself. The activity of learning to write computer programming code is a “hands on” activity that requires intensive focus and a huge amount of practice and training. According to Vihavainen et al. (2011), learning programming is very much about learning by doing. Students who have taken up programming in previous years will almost certainly have had more exposure to programming in a practical way, which is what gives them an advantage over students whose first experience in programming is at university.

Previous Experience and SE

Ramalingam et al. (2004) found that past experience is a strong determining factor of current SE and performance in computer programming. Their study indicated that students’ high school experience of programming created a sense of awareness of their programming abilities even towards the latter part of their university course. It was found that for a large number of students’ previous exposure to programming would yield high self-efficacy. According to Govender and Basak (2015), students who have previously taken a programming course, perhaps in high school had a significantly easier time reading and understanding the programming language as compared to first time programming students. Further evidence is provided by Kolar, Carberry, and Amresh (2013) who determined that students who have had previous knowledge or engagement with computing programming had a higher level of SE towards computing skills. It is thought that this might be because the students had good insight into what to expect as they were not seeing the material for the very first time. This shows that previous experience can positively affect self-efficacy as it gives students an idea of what to expect, which would increase their confidence going in to a computer programming course at a tertiary education institution such as a university or college. Students with minimal exposure to computer programming tuition may take longer to adjust and become comfortable with the type of content being presented. This can bring about negative associations with first time students as they might feel that they are slower at learning than their classmates.

School-based Computer Programming Experience

Armoni et al. (2015) determined that students who had previously learned programming in middle school could understand basic programming statements with minimal explanation needed from teachers in high school and they outperformed students in more difficult programming concepts such as loops. This outcome was refuted by Strong et al. (2017) in their study with 1st year university students where it was established that students who had previous high school experience with computer programming did not outperform students who did not have previous computer programming experience. Rather, students found that their past experience with programming gave them extra confidence when taking the class at tertiary level and they were more receptive to the learning process. Overall, they felt that having a familiarity with programming as a whole had a positive impact in their learning but did not necessarily have an impact on their skills and abilities. The outcome from this study is that previous experience has a positive influence on SE but not necessarily on academic performance because of a myriad of mitigating factors. The preceding assertion resonates with the outcome of a study that was conducted by Bennedsen and Caspersen (2005) where they found that students with previous experience relied too heavily on their past knowledge and eventually found themselves far behind in the course material. Students with this attitude are usually surpassed by students who had minimalist exposure to previous computer programming knowledge and mitigated for this shortcoming by making an extra effort to obtain mastery of the course work content pertaining to computer programming.

The Mediating Influence of Self-Efficacy

While previous experience has been acknowledged as a contributor to performance in computer programming, all evidence suggests that this is done through the mediating influence of self-efficacy. Hence, at the introductory level, the teaching of computer programming should focus on ensuring that students have a pleasant or positive disposition towards computer programming such that it sparks students' interest and passion for the subject thereby elevating their level of self-efficacy. Kittur (2020) found that the greater the programming experience of students the higher their level of programming self-efficacy. Students with more experience are also better at dealing with complex programming tasks. Kittur explained that these findings suggest that providing more experience to students and getting them involved with programming from their schooling years will support their SE in programming-related tasks. If students begin learning programming at school level they would be significantly more confident when they get to the tertiary sector. This confidence will also have an impact on their performance as discussed in the self-efficacy chapter. This point must be remembered because it will be not feasible in the South African context. Currently very few schools offer programming related instruction to pupils. Hence, the onus is on the tertiary sector to engage in innovative teaching methods that will enhance the SE levels of students and mitigate the lack of programming experience that most students suffer from.

Islam et al. (2019) conducted a study with students who had previous experience and two groups of students with minimal previous experience. It was established that students with minimal previous experience struggled the most with syntax and algorithms and students with previous experience had difficulties in debugging their code. In both groups the students demonstrated a preference for a YouTube tutorial or any video that contained actionable steps that enabled the completion of a programming task. This insight could be another possible recourse that programming teachers could use to help students complete an exercise successfully. Once students have followed along with a video tutorial a few times there can be a practical exercise where the students need to write a similar type of program on their own without a video assisting them. This process could, over time, also contribute to improving the students' self-efficacy. This is because completing programming exercises successfully, with the help of videos, will improve the students' confidence in themselves to successfully execute a program. This is a significant accomplishment that many beginner programming students have never been able to achieve, which leads to low self-belief.

Nazeri et al. (2018) suggest that students who have previous experience with programming either in secondary school or at home tuition perform better in programming classes due to already having exposure to problem solving, programming concepts and design concepts. Gathering these skills at an early age is beneficial for learning these concepts again at a tertiary level. Nazeri et al. (2018) also suggests that previous experience with mathematical classes also help students with programming later in their schooling years. Students'comfort levels when beginning a programming module also factor into the reason as to why previous experience plays a role in performance. Students who have taken programming as a subject at some previous stage of their lives tend to feel increased comfort levels when taking up a programming module at a tertiary stage. The experience however, is different for students who have had minimal prior experience with computerprogramming. These students will tend to feel a greater amount of anxiety when enrolling for a programming module and will not have any benchmark to assess the difficulty level of the module which leads to more feelings of stress and concern in these students (Nazeri et al., 2018).

However, according to Alexandron et al. (2012) past programming experience in some instances could cause misunderstanding of new concepts. It could also cause programmers to use the programming patterns in a way that they are familiar with from their previous experiences which leads to them missing out on better ways of creating programs. Alexandron et al. (2012) also found that when experienced programmers move from a detailed level of programming to programming at a high-level they tend to feel as if they have less control over the programming environment and this loss of control can lead them to develop a negative attitude towards the programming course. This development of a negative attitude towards computer programming tends to subsequently lead to poor performance at the subject itself. These findings suggest that students who have had experience with one type of programming paradigm or style may be thrown off if they enter university and find that they are being taught a different type of programming style. As an example, the change from procedural-style programming/functional programming to object-oriented programming tends to be quite a challenge. In such cases students who have previous experience may still perform poorly due to the different programming paradigms and environments being used or due to these students developing a negative attitude towards the subject because it differs from their preferences.

2.5 Intrinsic and Extrinsic Motivation

The construct of Motivation can be categorised into 2 broad categories named intrinsic motivation (IM) and extrinsic motivation (IM) (Ryan & Deci, 2000). IM refers to a person who is motivated to do something because they get enjoyment and pleasure from doing that task. EM means that a person is motivated to do something because of the outcome they will receive or to avoid a negative consequence. Gottfried (1985) found that intrinsically motivated students had more academic success than extrinsically motivated students. IM and EM are significant factors that determine performance in computer programming because of the inherent nature of programming itself (Tavares, Henriques, & Gomes, 2017). As previously mentioned, proficiency in computer programming is only achieved by virtue of having a very "hands on" attitude. The more a student practices and takes time to write programs the more they will become adept at it. This is why students who have a genuine enjoyment and passion for writing computer programs and designing software will vastly outperform students who are extrinsically motivated and only write computer programs because they are required to do so in order to complete an assignment or a homework problem. Furthermore, students who possess high levels of IM are thought to deal better with adversity and challenges that may arise when compared with students who are extrinsically motivated. Extrinsically motivated individuals tend to give up more easily when faced with difficulty or obstacles Rego, Sousa, Marques, and e Cunha (2012).

Factors that Determine Extrinsic and Intrinsic Motivation Traits

It is understood that numerous factors can determine whether a person is intrinsically or extrinsically motivated. However, in a study done by Kori, Pedaste, Leijen, and Tõnisson (2016) it was found that students who had previous exposure to computer programming before their first year at university had more intrinsic motivation than students who were exposed to it for the first time at university. Perhaps this was due to the knowledge that students who had previous experience with computer programming had navigated through the initial cognitive challenges that the learning on computer programming tends to present to the novice programmer and they were now in a position to leverage the enjoyable parts of computer programming to increase their levels of self-efficacy.

Furthermore, according to Forte and Guzdial (2005) and Kurkovsky (2006) students who study towards a non-computer science degree have less motivation to perform well in computer programming courses. This could be due to less exposure to computer programming course content and also, perhaps, the lack of a desire to work in the field of software development causes them to not take programming as seriously as students studying computer science. According to Bergin and Reilly (2005), a significant source of motivation for students in computer programming courses is eventually to become a professional software developer. For non-computer science students who are not driven by the desire to be employed in the software development sector, there is a substantial lack of motivation to acquire an expertise of computer programming content. The lack of IM leads

to reduced levels of enjoyment when it comes to computer programming thereby lowering students' SE and ultimately their performance in computer programming assessment.
Durak, Yilmaz, and Yilmaz (2019) also discovered that students' SE and motivation levels can be negatively affected by aspects of programming that they do not find enjoyment in such as challenging exercises and spending great effort trying to grasp a concept. However, their motivation can increase when with aspects of programming that they do find enjoyable. Most students in the study conducted by Durak said that in particular they enjoyed learning programming through using robotics as it felt like a fun activity rather than learning a skill.

According to the students, motivation levels tend to diminish when exposed to the more tedious and frustrating aspects of programming such as learning syntax and debugging their code. Furthermore, negative feedback on their progress and efforts from teachers or senior peers usually impairs students' intrinsic motivation and leads them to developing a negative attitude towards the subject itself. These findings suggest that students who may start off a programming course with intrinsic motivation may become exasperated with the parts of the subject that they do not enjoy as well as criticisms that they may receive from their senior instructors.

In a study that consisted of primary school students who undertook an 8-week course in IP, it was found that these students exhibited adult factors such as intrinsic and extrinsic motivation and previous experience had a distinctly positive effect on students' programming SE. In particular, students' inclination of pursuing a computer programming career later in life was found to be correlated to their level of intrinsic motivation for learning the subject matter. It was also found that students who had previous knowledge and engagement with computer programming displayed more extrinsic motivation than students with no previous experience (Aivaloglou & Hermans, 2019). The reason for this could be due to the increased levels of SE that students with previous experience tend to display, which may motivate them to spend extra time studying the material so that they are able to uphold their self-concept in this regard. According to Aivaloglou and Hermans (2019) the students in this study did not have stereotyped views of professional computer programmers as this study was conducted with students of a young age who were not yet exposed to these biases.

Yacob and Saman (2012) found that programming students had two main sources of intrinsic motivation which were attitude and setting themselves stimulating goals. The aspect of attitude tended to come from a student's prior experience with programming and whether or not the current programming content that they are learning meets their expectations. The extrinsic factors that were found to motivate students the most were "clear direction, reward and recognition, punishment and social pressure and competition." Each of these factors were found to positively contribute to student's motivation to engage with the programming content. Out of each of these motivating factors, the main contributors were the desire to obtain a precise direction in terms of what was required, the consequences of failure and what reward was on offer if they achieved success. The factor of setting stimulating goals had the lowest motivating outcome from all of the factors. Yacob and Saman (2012) added that teachers of programming courses should try to communicate programming tasks and assignments as clearly as possible so that students can easily find their "clear direction" which would help motivate them in completing course work.

Khaleel, Ashaari, and Wook (2019) conducted a quantitative study with how gamification affects students' motivation and performance in programming. Gamification entails using a game style approach to learning something difficult like programming to increase motivation and engagement of students Khaleel, Ashaari, Wook, and Ismail (2015). Gamification uses techniques such as scoring points, earning badges, leader boards amongst students and team activities to drive engagement in a subject (Elshiekh & Butgerit, 2017). According to Khaleel, Ashaari, and Wook (2020) using gamification methods such as badging can push students to complete homework or programming assignments so that they can get the immediate reward of earning a badge. Leader boards can show students their class ranking amongst other students which can act as a form of social pressure that incentivises them to improve their ranking. An experiment was conducted with 90 Information Technology students at a University. The students were separated into a group that learned programming using gamification and a group that learned programming conventionally. The evidence from this study indicated that there was a significant difference in motivation levels between the 2 groups. The students in the experimental group reported having greater motivation to learn the programming material than the students in the other group. According to Khaleel et al. (2015) gamification exploits the extrinsic motivation that all humans naturally possess and uses this motivation to make learning less boring and more satisfying and rewarding. 80% of students were quoted as saying that they would enjoy their tertiary studies more if it included game-like elements in the courses, and 60% of students said that their motivation would increase if their University displayed leader boards as this would encourage more competition between their peers and themselves (Andriotis, 2014).

2.6 Problem Solving Ability

Heppner and Petersen (1982) suggest that problem solving is the activity of achieving a goal when the method of achieving that goal is uncertain. According to Balmes (2017), students who achieve high scores in mathematics tend to also do well in computer programming assessment. This phenomenon can be as a result of the problem-solving abilities required in mathematics being very similar to the cognitive skills required in computer programming. Balmes goes on to suggest that excellence in mathematics is an indicator of a student who has the cognitive ability to learn computer programming. A study of students attitudes towards mathematics was conducted by Ali, Ali, and Farag (2014). The study found that students' attitudes towards mathematics were significantly correlated with their performance in computer programming assessment. In the Balmes study it was discovered that mathematics scores can be a predicting factor in whether or not students will be able to pass university programming courses.

Computer Programming and Mathematics

A study conducted by Duran (2016) on university students similarly found that students computer programming marks were similar to their course marks in maths. These results are not surprising because mathematics is underpinned by rule-based logic and so is computer programming. Mathematics is driven by theories that are integral to problem solving. This is identical to computer programming where the syntax of a computer programming language coupled with the semantics of the logical rules and data structures provide the theoretical foundation for problem solving.

Based on personal experience, the researcher realised that the logic of the computer problem solving process entails the development of solutions that require the designing of an interface, the writing of lines of code that align to a solution that has been logically designed and the integration of this solution with database technology. An IT degree provides a student with an ability to utilise computer programming knowledge and develop a fully-fledged solution to a societal or business problem. According to Lishinski, Yadav, Enbody, and Good (2016), there is a correlation between problem solving ability, the mental model of the problem domain and computer programming performance. However, this correlation did not apply to knowledge of simple programming structures and superficial mastery of computer language syntax. It applied more significantly to the advanced aspects of computer programming where students had to develop a fully-fledged computer programming solution to a real-life problem.

2.6.1 A Mental model visualisation of the problem domain

Du Boulay (1986) suggested that the greatest impediment to successful acquisition of computer programming skills was the issue of mental model. A mental model is the mental image that students construct of the problem domain and it is within this image that students are able to foresee a solution. However, this solution path is also influenced by *the problem of orientation.* This aspect deals with students' overall understanding of what computer programming can accomplish and why it is important to gain expertise of the activity of computer programming so that the computer-oriented solution can be developed. Du Boulay conflates the concepts of the mental model and the issue of orientation to what is referred to as the *notional machine*. This is the challenge that students are faced with when they do not fully understand the inner workings of the computer as a machine and how this relates to the writing of programs that represent a solution to a problem. Students need to grasp the concept that a machine will only understand and execute coding statements if it is written in a certain way and in a specific order, also referred to as the syntactical rules that are imposed on the computer programmer. Also, it is important to comprehend how the computer as a machine will store data such as inputs and outputs as well as data structures so that the manipulation of this data can be easily achieved. Many students do not understand this relationship between the code from a logical perspective and its physical implementation on the machine. This leads to a layer of abstraction that makes computer programming very difficult to understand. Such an appreciation for the notional machine emanates from a deep and meaningful introduction to the activity of computer programming. However, many educational institutions do not have the latitude of embracing time-consuming pedagogical strategy because of the imperative to cover course content in a confined time period. The challenge that academics are faced with is how to integrate the concept of the notional machine into course content in a seamless manner that enhances students' mental model of the problem domain. The role played by the syntax and semantics of computer programming code was further explored by I. Govender (2021) who conducted a phenomenological study on the difficulties that novice programmers face when they learn to program. In order to enhance the mental model visualisation of the problem domain, I. Govender (2021) stresses the importance of using a "scaffolding" approach to the teaching of computer programming that entails a strong focus on baseline knowledge involving computer programming language syntax and an inculcation of a deep appreciation for data types. Once these fundamentals have been entrenched, students will be cognitively prepared to engage in incremental learning that involves algorithm development and problem solving.

2.6.2 Formal Language Notation and Problem Solving

Du Boulay also refers to *notation of formal languages* as another aspect of difficulty. Novice programmers often have difficulty in remembering the syntax of the different programming languages they are required to learn (which concur with Butler and Morgan (2007)). This handicap is often coupled with an inability to understand the semantics of language constructs. The semantics of computer programming languages manifest in computer programming structures such as selection statement (commonly referred to as "if…then" statements), looping structures and data structures that are used to temporarily store data for quick and easy manipulation. Added to this mix are the complexities inherent in learning object oriented (OO) programming techniques such as the concept of a constructor, the use of inheritance hierarchies, polymorphism and method overloading. The challenge of *acquiring mastery over structures* is a constant struggle for both teachers and students. Data structures in programming are usually complex to teach to students on a theoretical level and equally difficult for students to understand without first working through numerous practical examples of these structures. These challenges have been documented by studies such as: Robins, Haden, and Garner (2006) who highlight the difficulties faced in learning looping and arrays; Goldman et al. (2008) who note the problems students have with learning inheritance and Garner, Haden, and Robins (2005) who allude to the abstractionism inherent in understanding how a constructor instantiates an object of a class.

2.6.3 The Pragmatics of Computer Programming

Finally, Du Boulay (1986) mentions the "*pragmatics of programming"* as a source of difficulty to many novice programmers. The pragmatics refer to aspects such as understanding what needs to be accomplished to develop a solution that meets the requirements of a problem, handling errors that may arise, debugging the program and time management. Beginner programmers are ill equipped to handle the realities of constructing a proper solution to a problem and may only become aware of these issues once they have started working on the task. Usually problems like these are enough to derail a whole project for a beginner, which leads to students losing morale and motivation in learning computer programming all together.

Hence, a student’s problem-solving ability, as well as their mental model visualization of the problem domain is a significant predictor of computer programming performance.

Authors such as de Araujo, Andrade, and Guerrero (2016) and Romero, Lepage, and Lille (2017) have stressed on the importance of problem solving ability as a crucial factor in enhancing algorithmic thinking capacity. This assertion is supported in 96% of the papers that they reviewed as part of their systematic literature review. Wing (2008) defined algorithmic or computational thinking as “the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent” (p. 3717).

Lawan, Abdi, Abuhassan, and Khalid (2019) conducted a study with 113 Information Technology and Engineering students at a university and it was found that when students spent additional time on improving problem solving skills it had a positive effect on their perceived ability to learn programming. Students with better problem-solving skills perceived learning programming languages as easier and encountered fewer difficulties with learning the programming language as students who did not perceive their problem-solving ability as highly. In this study it was also concluded that demographics such as age, race and academic department do not have an effect on students’ problem-solving ability or their ability to learn programming.

In a study conducted by Ismail, Ngah, and Umar (2010) that comprised of interviews with 5 computer science lecturers at a university about the reasons for students’ poor performance at programming courses it was found that all 5 lecturers agreed that students tend to lack the fundamental skills needed for analysing and solving a problem. The lecturers believed that the reason for this lack of skill is that students were not taught how to think logically and dynamically at any point in their lives. It was suggested that students should be required to acquire logic, problem solving and creative thinking skills, possibly through a mathematics module, before registering for a programming module. It was also found that students often-times do not have the required problem-solving ability along with not knowing the syntax or how to write a programming statement correctly. This compounds on students’ problem-solving challenges because even if they do eventually understand and breakdown a problem correctly they cannot ultimately write code that works. It was agreed that students should spend more time actively engaging with the tutorial and practical sessions and they should also receive feedback on their performance in these areas so that they can find their gaps in knowledge (Ismail et al., 2010).

A preliminary study was conducted by Bain and Barnes (2014) which questioned students on the challenges they experienced when they learnt how to write computer programming code and it was established that 50% of students did not have a strategy of dealing with problems that arose while writing computer programming code. The main method of trying to solve the problem was to turn to internet searches. It was also gathered that 53% of students did not understand how different programming concepts and elements of code related to the bigger picture and how small sections of programming topics connected with others to form a whole solution to a problem. It was concluded that the fundamental issue with learning programming was inadequate problem-solving methods and a lack of critical thinking.

Loksa et al. (2016) undertook a study of 48 high school students who attended a website development camp for two weeks. The researchers conducted experiments where the students were taught the basics of problem-solving and were exposed to strategies that programmers typically use to solve problems and tutored on how programmers decide on the correct path towards a solution. Two web development camps were set up and the programming performance of students in each camp was measured. The study was set up as an experiment where one camp had received the problem-solving training and the other camp did not. The students were given a project of developing a web application within a duration of two weeks. The main finding from this study was that the group of students who were tutored in problem-solving performed better due to higher rates of productivity when working on their websites. They also requested assistance from teachers less frequently and generally tried solving an issue by themselves before requesting for assistance. The students who received training were also found to have higher levels of SE and had adopted more of a growth mindset towards web development which essentially means that they saw the web development task as something they can learn from and improve at over time. Loksa et al. (2016) gives the recommendation that before students begin to engage in computer programming assessment tasks they first need to have foundational knowledge of problem-solving strategies and procedures that they can follow once they begin with development.

2.7 Deep and Surface Learning

The approach that students adopt towards learning has been a topic of vast research pertaining to tertiary education (e.g. Lonka, Olkinuora, and Mäkinen (2004); Vanthournout, Coertjens, Gijbels, Donche, and Van Petegem (2013); Trigwell, Prosser, and Waterhouse (1999)). According to Marton and Säljö (1976) the main approach to learning is framed by concepts referred to as deep and surface learning. This was introduced in their seminal study where they gave students a passage to read and then asked the students to explain the main ideas presented in the reading and asked them what approach they took to the reading task. From the results of the study they found two different types of students. Students who wanted to gain an understanding of the information provided in the reading and students who wanted to remember information for the sake of being able to reproduce it in at the end of the session. From this, Marton and Säljö introduced the concepts of a deep learning approach and a surface learning approach. They found that students who were genuinely interested in their classes and tried to obtain a genuine understanding of the academic material had a deep approach to learning. This type of student attaches personal value to the concepts and knowledge gained in class. Students who on the other hand use memorisation and rote learning techniques rather than understanding to pass tests and exams are said to adopt an approach referred to as surface learning (Spada & Moneta, 2012). Students who have a surface approach to learning may be able to pass and even excel in a subject, however their learning style is only appropriate in test and examination situations where they are required to simply reproduce information but in situations where they are required to use this information in a practical way they usually fall short. According to Lindblom-Ylänne, Parpala, and Postareff (2018) students using a deep approach to learning apply critical thinking skills, thereby enabling them to make connections between different concepts more easily. Lindblom-Ylänne et al. also went on to mention the link between the surface learning approach and self-efficacy. Students with low SE, low motivation to study and negative beliefs about studying tended to use the surface approach to learning. This again reiterates the importance of SE and motivation on students and the effect that this mindset has on students and their learning abilities.

Hulleman (2007) found that when students were asked to apply programming concepts to their personal lives the students started to develop more of an interest in the programming at school and began performing better at the subject. The reason for this according to

Hulleman (2007) is that the students began to see more value in the work they were doing which made them care more about learning these concepts to understand them. This effect was found to be more pronounced in students with low SE.

According to Floyd, Harrington, and Santiago (2009) students who participate more in class activities and adopt a positive attitude towards computer programming will tend to engage in more deep learning techniques. In the same study it was found that students who did not adopt negative perceptions of the course engaged in surface learning techniques. Floyd et al. (2009) says that the reason for this is that surface learning is a "survival strategy" for students when they do not find a course engaging and meaningful to their lives.

According to Jenkins (2002) computer programming is more complex of a subject than most other subjects based on theory or rationale because programming consists of both. It requires learning of concepts through rote memorisation techniques while also having a deep understanding of code and practically working with developing programs to master the skill. He argues that this is where the problem with learning programming lies as students are unfamiliar with applying this blend of deep and surface learning as it is not commonplace in most other subjects.

Biró and Csernoch (2014) found that by making use of surface learning strategies students are only storing certain parts of programming logic in their short-term memory. The understanding of why the logic works and how to apply it in different scenarios is not developed in their minds which is why it does not transfer into their long-term memory and this is why students who use rote memorisation techniques for concepts like programming functions often fail to reproduce the function correctly and also fail to adjust and change the function to fit the context of the question in examination situations.

Fincher (2006) discovered that students who adopted a deep learning approach try to find value by relating the subject matter to their personal lives and finding their own importance in learning the material. This also suggests that students with a deep learning style are generally intrinsically motivated. On the other hand taking a surface learning approach is linked to more extrinsic motivations such as avoiding failure or social pressure. In a study where that entailed the interviewing of 177 university students who enrolled in a programming module, it was established that students who scored high on deep learning attributes also achieved high marks for their computer programming module (Fincher, 2006).

In a study conducted by Hughes and Peiris (2006) a questionnaire was administered to a class of computer science students at a tertiary institution. The results from the questionnaire were assessed alongside the students marks in their programming assessments. The researchers found that students who adopted a surface learning approach performed the poorest. It was also found that students who took a deep learning approach performed better, however, not as well as students who took a strategic learning approach. A strategic learning approach is when students learned programming with the intent of achieving high academic marks. The reason students who took a strategic approach could have performed better may be because these students adopted both deep and surface learning techniques to both memorise the concepts and gain an understanding of how to apply the concepts to different scenarios. It was argued that the students who adopted a fully deep learning style may lose sight of how to apply programming concepts in testable situations as they become too absorbed in developing a personal interest towards it (Hughes & Peiris, 2006).

According to Peng, Wang, and Sampson (2017) students can be encouraged to engage in deep learning strategies by being assessed through project work instead of only being assessed through written examinations. By allowing students to work on a project such as developing a web application, either individually or in a group, the students will find more meaningfulness in the subject matter as they become invested in their projects. Peng et al. (2017) also suggests that through project work educators can better monitor and provide assistance to students and identify their weaker areas more efficiently. Konecki and Petrlic (2014) agree that programming needs to encompass both deep and surface learning approaches because of the fact that programming is more of skill than knowledge.

According to Malik, Shakir, Eldow, and Ashfaque (2019) by teaching students problem solving skills and strategies this will inherently promote students to adopt deep learning techniques because being able to analyse a problem and converge at a solution is the same as taking on a deep learning approach.

According to Ranjeeth (2011) 50% of computer programming students at tertiary education institutions have a tendency to adopt a surface learning approach for computer programming in introductory courses. The researcher suggests that students tend to adopt this style of learning to meet the course requirements and to be able to obtain a pass mark for programming assessment. This strategy results in the acquisition of a superficial understanding of computer programming that usually manifests in the final year of study where the level of computer programming knowledge required to pass courses does tend to become more intense.

Hence the adoption of deep and surface learning towards computer programming does become a factor that needs to be examined in greater detail in terms of its influence on students' performance in computer programming assessment.

2.8 A Reflection of the Literature Review

While there has been numerous other studies on the individual factors that affect academic performance in computer programming by students, there is a lack of academic studies that have focused on the combination of factors such as mental model visualisation of the problem domain, psychological factors (intrinsic and extrinsic motivation as well as self-efficacy), previous experience and learning style/approach. Research efforts on these factors have been quite disparate with studies tending to focus on a single factor or just a few factors. Based on the literature review, the number of factors that influence performance in computer programming have been saturated and have converged to a finite manageable list. The study's conceptual model was leveraged on the knowledge that the constellation of factors that have been integrated into the framework have all been recognised and endorsed by the community of academics who have made contributions towards understanding the phenomenon of student performance in computer programming. An immediate outcome of the literature review was that it provided a context that forms a suitable backdrop to the presentation of the study's conceptual framework.

2.9 The Conceptual Framework

The study's conceptual framework was constructed on the basis of the factors that have been hypothesised to influence performance in computer programming. These factors are illustrated in Figure 2.1.

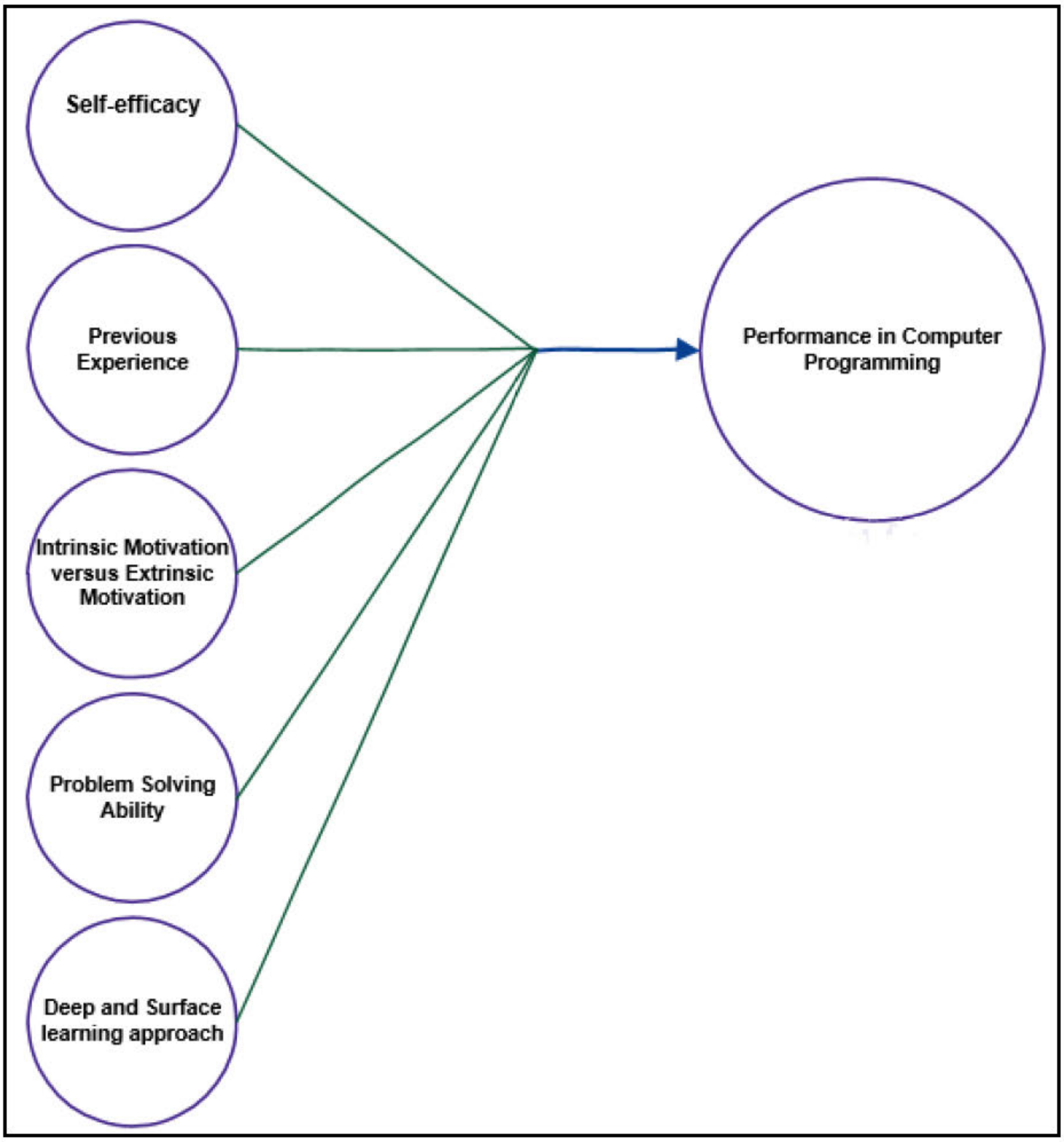

Self-efficacy
Previous Experience
Intrinsic Motivation versus Extrinsic Motivation
Problem Solving Ability
Deep and Surface learning approach
Performance in Computer Programming

Figure 2-1: Factors that Influence the Learning of Computer Programming

Figure 2.1 represents a primitive arrangement of factors that could be further arranged according to dependent and independent variables as illustrated in Figure 2.2. The factors identified in Figure 2.1 were taken from studies conducted by Bandura and Wessels (1994); Duran (2016); Fang (2012); Gottfried (1985); Kori et al. (2016); Spada and Moneta (2012).

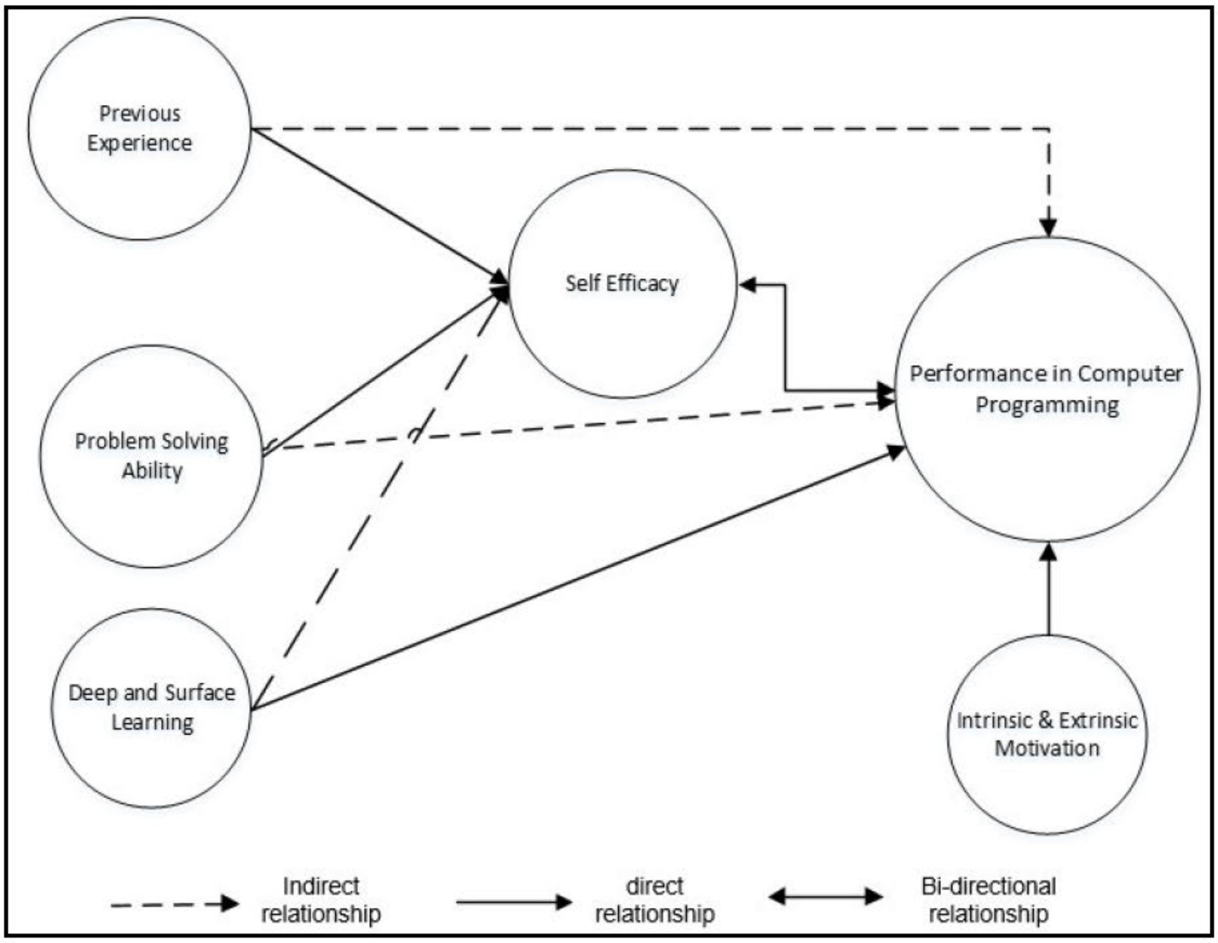

Previous Experience
Self Efficacy
Performance in Computer Programming
Problem Solving Ability
Deep and Surface Learning
Intrinsic & Extrinsic Motivation
Indirect relationship
direct relationship
Bi-directional relationship

Figure 2-2: A Conceptual Framework for the Learning of Computer Programming

In Figure 2.2, the independent variables are Previous Experience, Problem Solving Ability, the Learning Approach (Deep and Surface) and Intrinsic and Extrinsic Motivation. According to the discussion in the literature review, previous experience and problem-solving ability have a direct influence on a student's self-efficacy towards computer programming which manifests on a student's ability to learn computer programming. The learning approach adopted by a student in terms of deep and surface learning also has a direct influence on academic performance in computer programming as do intrinsic and extrinsic motivation. While it has been established that previous experience and problem-solving ability have an influence on performance of computer programming, these influences are mediated by self-efficacy.

The dependent variable in the study is the students' proficiency/academic performance in computer programming. This variable was measured by obtaining a self-assessment-based rating of students' performance in computer programming assessment. The researcher was aware that students in the IS&T department at UKZN engaged in a formal practically-based

computer programming assessment where they were required to use their programming skills holistically to display their proficiency of computer programming and provide a successful solution for the task given to them. It was envisaged that the mark obtained by the students would provide a guideline to enable the students to rate their individual performance in computer programming assessment. This self-reported rating will be used as an indicator of the students' academic performance in computer programming. The strategy of using practical computer programming assessment activity as an indicator of proficiency in computer programming has also been used in studies with a similar agenda as the current study (e.g. Bennedsen and Caspersen (2007); Edwards, Murali, and Kazerouni (2019). The preceding discussion reflecting the link between the dependent and independent variables in the study is reflected by the illustration in Figure 2.1.
According to Abdunabi et al. (2019) self-efficacy in computer programming is theoretically linked to a students' background and previous exposure to programming as well as their background in mathematics and problem solving. This link assumes that students who have a good background in mathematics and have been educated in some form of computer programming content (algorithmic or language syntax), prior to attending university, tend to have a higher self-efficacy (SE) in programming. This is based on the knowledge that these students have constructed a foundation of knowledge that assists them in understanding programming concepts at university, regardless of the programming language being taught. A students' level of SE then has an impact on their learning style as students with higher SE are more likely to adopt a deep learning style as they tend to find the subject inherently interesting. Finally, the overall combination of each of these factors, self-efficacy, deep and surface learning, previous experience problem solving ability and intrinsic and extrinsic motivation result in a student achieving higher marks in programming tests and exams. The results of tests and exams then feeds back into their self-efficacy, if they have performed well in a test or exam this will work to increase their belief in themselves their programming abilities which then results in them consistently performing well on tests and exams. According to Yacob and Saman (2012) both intrinsic and extrinsic motivation have a positive relationship to the learning of computer programming. Also, students who find the subject more enjoyable will develop both intrinsic and extrinsic motivation to work on programming tasks, thereby ensuring that they are adequately prepared for exams and assignments pertaining to computer programming.

2.10 Virtual Learning due to Covid-19

The Covid-19 pandemic has profoundly impacted on teaching and learning platforms globally. The adoption of online learning has changed the learning behaviour patterns where students have been forced into isolated learning situations. This resulted in a pedagogical shift from on campus learning to virtual classrooms for programming students across the globe (Mbunge, Fashoto, & Olaomi, 2021). This shift in learning was envisaged to possibly manifest as an additional layer of complexity to the current study. However, the academic literature on this topic does not provide a consensus on the role that online learning has on academic performance in computer programming, as indicated by the following discussion.

In a study conducted with a group of 45 first year university students studying object-oriented programming using C++, Maltby and Whittle (2000) found that having face-to-face (F2F) interactions with lecturing staff, tutors and other students did not have a significant effect on the final outcome of students marks in programming exams compared to online tuition. The most significant determinant of success in programming exams was the level of effort that an individual student applied to their programming material. This result was in contrast to what many students thought determined their marks which was attending physical classroom lectures and interactions with educators and tutors. The results of the study also showed that the students who generally performed well in C++, before moving to virtual learning, still continued to perform well after moving to online classes. These findings are supported by other research such as a survey done by Co and Chu (2020) that found that 73.4% of students did not find online learning any more easy or difficult than face-to-face learning.

However, in a study with a similar agenda by Khraishi (2021), it was reported that more than 70% of students agreed that online learning (OL) may have advantages over traditional on campus learning. A study by Li et al. (2021) looked at a group of programming students who participated in pair programming sessions through Zoom and found that students derived large amounts of value from having these sessions as they were able to problem solve more collaboratively with their partner, brainstorm different ideas and impart/gain more knowledge when it came to writing code. While this study provides evidence that OL from a computer programming perspective is both feasible and practical, a study by Chen, Lasecki, and Dong (2021) reported that a majority of the students (58%) had a preference for traditional face-to-face lectures over an online mode. The main reason for this phenomenon is based on student perception that the traditional (face to face) teaching and learning style offers greater educational value and provides more opportunity for meaningful engagement with academic staff. A significant outcome from this aspect of the report from this study is that a majority (70%) of high-achievers and a general majority (56%) of students were of the opinion that the most problematic aspect of learning computer programming via an online mode was the ability to understand aspects of computer programming that were deemed to be cognitively challenging. This includes aspects such as looping, selection structures, object-orientation (OO) and data structures. The ironical outcome from this discussion is that the perception-based evidence did not match the empirical evidence regarding students' performance in computer programming. The empirical evidence attested to by students' academic performance in computer programming assessment indicated that the high-achieving students performed equally well with OL as they did with face to face (FTF) learning. This outcome is not a generalised one as the analysis of results of low-end achievers did not provide a significant correlation between FTF and OL.

The conflicting reports regarding the role played by OL versus FTF learning provides evidence that the introduction of OL as a variable in the current study is not warranted at this stage of such a study.

2.11 Conclusion of the Literature Review and Conceptual Model

According to Levy and Ellis (2006), a crucial reason for conducting a literature review is to establish what has been discovered in the field of study and to provide a context for the intended study. The literature review section for the current study has been designed along the dictates of Levy and Ellis and the progression of the literature review has been guided by the academic sources on the topic. The literature review has been designed to provide an exhaustive coverage of the main topics that prevail in this domain of study. The main coverage areas are illustrated in the hierarchical diagram in Figure 2.3

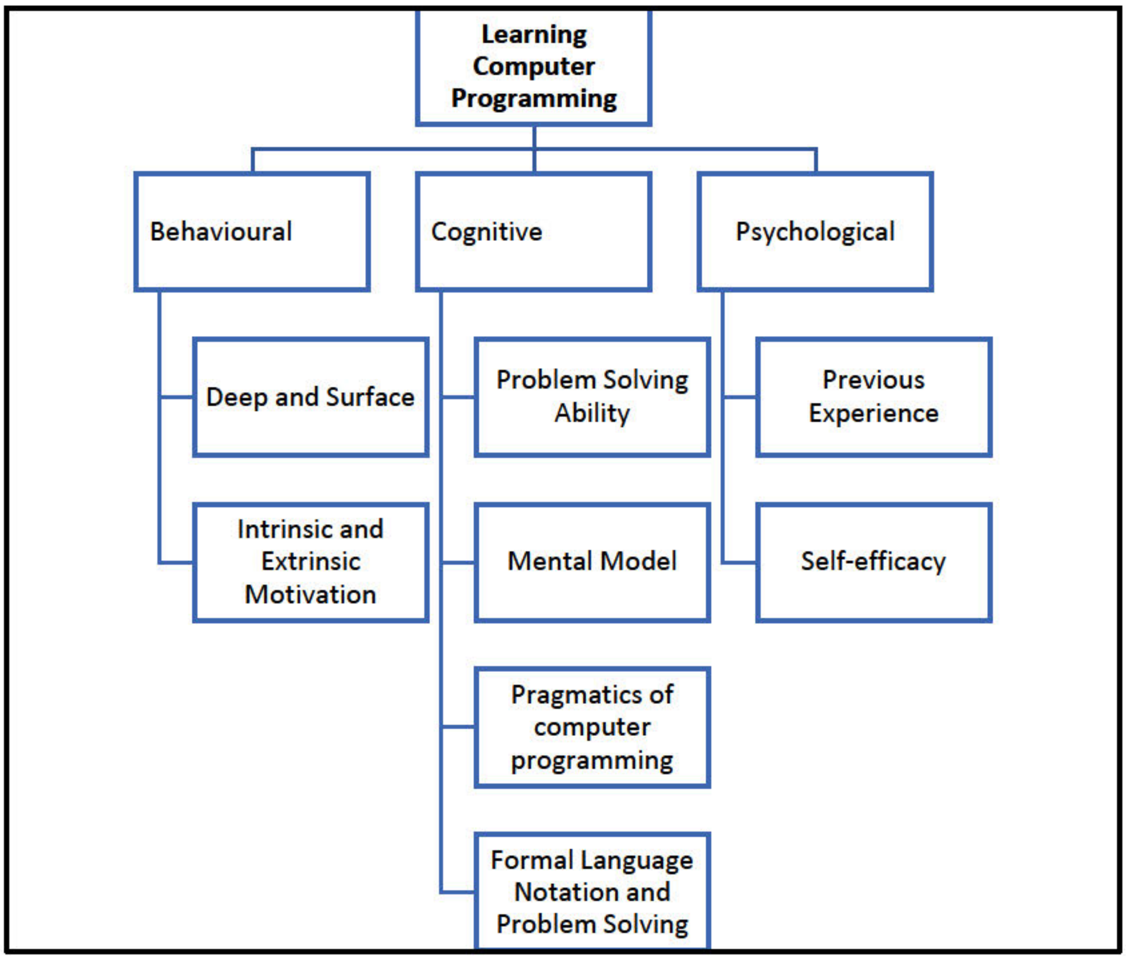

Learning Computer Programming
Behavioural
Cognitive
Psychological
Deep and Surface
Intrinsic and Extrinsic Motivation
Problem Solving Ability
Mental Model
Pragmatics of computer programming
Formal Language Notation and Problem Solving
Previous Experience
Self-efficacy

Figure 2-3: A Hierarchical overview of the Literature Review

The broad classification of topics covered in the literature review provided a foundation from which the researcher was able to develop a conceptual model to guide the data collection phase of the current study. The conceptual model (Figure 2.2) provided the researcher with a platform from which the study's research design and data collection instruments were developed. These core constructs from the study will be discussed in the next chapter. The literature review also refers to the strategy of online learning and its influence on computer programming performance. The outcome of this inquiry indicates that online learning (OL) did not significantly influence academic performance in computer programming. Informed with this knowledge, the researcher continued with the study by not directing any focus on online learning as a variable that could have influenced the outcome of the study.

CHAPTER THREE – The Research Methodology

3.1 Introduction to the Study's Methodology

A pre-cursor to a study's research methodology is the research design and according to Sekaran and Bougie (2016), there are many aspects to be considered. These aspects will be used and discussed in accordance with the Research Design Framework provided by (Sekaran & Bougie, 2016, p. 102).

The main purpose of the current study is to examine the relationships between the main variables of the study's conceptual model so that a more nuanced understanding of the factors that influence computer programming performance could be established. This will enable the researcher to use this empirical evidence to answer the study's cohort of research questions. The type of investigation is classified as a correlation-based study with the objective being to establish whether there is a statistically significant correlation between the study's main variables. The extent of the researcher influence in the empirical phase of the study will be minimal and the planned data collection from a time-horizon perspective will be cross-sectional. It is suggested in Sekaran and Bougie (2016) as well as Mark Saunders (2011) that a viable data collection approach for a cross-sectional study will be via the method of surveys, interviews or a combination of both. Many of the research methodology texts (e.g. Creswell & Creswell, 2017; Mark Saunders, 2011; Sekaran & Bougie, 2016) have identified 3 primary approaches to conducting an empirical study. These are the quantitative, qualitative and mixed methods approaches. While the research design is meant to be the overall plan that guides the study and ultimately defines the methodology, it is the researcher's worldview orientation (Saunders, Lewis, & Thornhill, 2009) that plays a significant role. The researcher's worldview orientation is a reference to the researcher's epistemological and ontological perspective on how new knowledge is obtained. The researcher currently has a strong leaning towards an objectivist worldview that is defined by unobtrusive data collection and analysis. However, in order to establish whether such a worldview orientation will be ideal for the current study, the researcher will provide an overview of the various research approaches and based on an analysis of each, a specific orientation will be selected and used to define the methodological detail for the current study.

3.2 The Quantitative Research Approach

According to Apuke (2017), quantitative research is focused on gathering numeric data and analysing that data to find statistical relationships, which can help the researcher test a hypothesis. The main method of gathering data is through distribution of a questionnaire which is filled out by a set of participants to determine their thoughts or experiences regarding a particular research topic. Quantitative research is underpinned by positivism, which argues that an objective reality exists and is detached from the human mind (Rahman, 2020). Some of the key advantages to quantitative research is that it can depict an objective stance on the research problem. It can also be generalisable to other populations because it uses a large sample from which insights can be gathered. It is also one of the least time consuming methods of acquiring data (Rahman, 2020).

Law, Lee, and Yu (2010) conducted a study to determine how motivation influenced academic performance in computer programming by students. The researcher took a quantitative approach by making use of a questionnaire that was distributed to a group of tertiary programming students. In total the researcher had collected 365 valid questionnaires filled out by students. From the results obtained the researcher was able to attribute a correlation between the students who were intrinsically motivated and their enjoyment of programming. It was also found that students who had intrinsic motivation reported greater self-efficacy in programming than students who were extrinsically motivated.

According to Korkmaz and Altun (2014), the levels of self-efficacy displayed by students in a computer engineering course was greater than that of general engineering students. The reason for this difference is determined to be that computer engineering students are exposed to more programming as they enrol for more programming classes at university and are therefore exposed to more programming related assignments and homework as opposed to general engineering students who take more engineering related classes at university. This is an indication that one's amount of previous experience in programming leads to higher self-efficacy. This study was underpinned by a quantitative survey that was carried out amongst 378 computer and general engineering students. The aim of the study was to determine students' self-efficacy in a C++ programming class that both groups of students had completed.

3.3. The Qualitative Research Approach

According to Alase (2017), qualitative research allows for different individuals to communicate their experiences of a particular event or aspect of their lives. Information was gathered through listening and documenting individual experiences as well as their thoughts and opinions of what they experienced. In the context of computer programming, qualitative studies have achieved moderate success in understanding difficulties associated with academic performance in computer programming.

In a qualitative study conducted by Lishinski, Yadav, and Enbody (2017) with a group of university students learning programming, the study attempted to collect statements from each of the students after a programming assessment had taken place to understand their feelings and emotional state. The results found that students previous performance had a large influence on how they felt about programming assessments and their overall self-efficacy. They also found that students who felt negative emotions because of performing poorly in previous assessments tended to have a low self-efficacy and this continued to affect how they perform in programming. The findings also suggest a feedback-loop that results in students having a negative outlook on programming from previous experiences and then this causes a lack of interest and motivation for the subject, which contributes to their low marks in future tests and assignments. The qualitative nature of this study allowed the researchers to delve deeper into subjective experiences of students, which as the researchers prove can yield significant results and patterns. The main benefit of adopting a qualitative approach is that it allows the researcher to adopt a deeper understanding of specific aspects of the task of learning to program; However, the generalisability of this methodology has been flagged as one of the main challenges (Ochieng, 2009).

3.4 The Mixed Methods Research Approach

According to Hanson, Creswell, Clark, Petska, and Creswell (2005), the mixing of qualitative research methods with quantitative methods is named the mixed methods research approach. The idea behind this methodology is that the strengths of both paradigms of research are leveraged to produce a well-informed research outcome. In the context of learning of computer programming, mixed methods research has been widely used.

In a paper done by Mather (2015) to study how students learn the C programming language, the researcher employed a mixed methods approach, which consisted of distributing questionnaires to students and reviewing their marks on a programming test as the quantitative component, and observation of participants through video analysis as the qualitative component. Through this study the researcher found that students performed better when working with other classmates in pair programming activities, which was confirmed by students' responses in their questionnaire showing their preference for collaborative working sessions. During their time spent in pair programming students were able to accomplish more in a shorter space of time and the researcher observed this through the video analysis portion of his study. However, from an analysis of the students' test scores the researcher found that students collaborating did not help them with remembering key programming concepts in the test as reflected by their scores. The researcher concluded that working collaboratively does not equate to deep learning, which is necessary for performing well without any assistance. These findings required multiple data collection methods in order to reveal different truths.

Another mixed-methods study conducted with 49 primary school students learning programming for the first time on the Scratch programming platform conducted focus group interviews about the students perceptions of programming and also did a quantitative analysis of their programming test results (Kalelioglu & Gülbahar, 2014). The study found through the student interviews that students had a positive impression of Scratch programming, with many of them stating that it was an enjoyable experience. After spending some time learning Scratch the students also said that they felt more confident with their programming abilities. While the qualitative results showed positive results overall the analysis of the students marks and tests showed that many of the students still struggled with their problem-solving abilities and this was a key area of weakness that the students still needed time to develop. The qualitative results did support these findings as many students did admit that they had a low perception of their ability to problem solve but still found enjoyment in the programming activities regardless of the difficulty they experienced with some of the questions. This is another example of how the mixed-methods approach can reveal more substantial information pertaining to different factors or aspects of how students learn programming. It also shows how different data can be used in tandem as supporting evidence.

In another study Zahedi et al. (2021) conducted a mixed-methods study to determine if gamification had an influence on female programming students. From the quantitative results it was found that female students benefited from gamification practices just as much as male students, and this was found through analysis of the students' results. However, these results were in contrast with the qualitative results. In interviews, most female students said that they were apathetic or felt negatively to some gamification practices, while some female students felt that gamification actually reduced their motivation towards the subject. These conflicting findings allowed the researcher to question more deeply if it was actually the gamification practices that had a positive impact on students marks or if other factors such as intrinsic motivation and self-efficacy could have played a larger role.

Hence, while the mixed methods approach does provide a greater opportunity for engagement with the respondents of a study, there is also a huge prospect of obtaining conflicting outcomes because of the fundamental difference in methodologies.

3.5 Justification for the Quantitative Methodology

After reviewing the main approaches to an academic study, the researcher gravitated towards a quantitative methodology to underpin the study. This decision was based on the observation that many of the correlation-based studies on the factors that influence academic performance in computer programming has been conducted using a quantitative approach that manifests in a survey type of methodology. This survey type of approach has also been conducted quite successfully in the studies that are detailed in the subsequent discussion. Many of these studies have focused on constructs that are core to the current study. As an example the construct of self-efficacy was studied by Karsten, Mitra, and Schmidt (2012) with a sample of 151 IT teachers before and after they had completed an online course on Scratch programming to determine if learning Scratch programming had an impact on their programming self-efficacy. The study was conducted using web-based questionnaires with Likert Scale questions about programming self-efficacy and perceptions as well as questions about the participants attitude towards programming. The results of the study found a *significant* difference in self-efficacy and attitude towards programming after completing the Scratch course. It was found that the participants were able to think more positively about programming and this influenced their belief that they could handle more long and complex programming tasks in the future. These results show that some of the main factors discouraging people from taking an interest in programming is a negative attitude and lack of confidence and motivation (Karsten et al., 2012).

In a study to analyse the factors that *motivate* students to study computer programming Law et al. (2010) distributed questionnaires to students who were enrolled in two different computer programming courses. The questionnaire was distributed to 386 students in total and of the questionnaires returned 365 were valid samples that were used in the analysis for the findings. The study found that factors such as individual attitude, having a clear direction, and being rewarded for good work had the largest impact on students' motivation to learn programming. It was also found that students' who set challenging goals for themselves and students' who felt social pressure to compete with their classmates had greater levels of motivation. A smaller study on motivation and learning programming was conducted by Yacob and Saman (2012) with 30 valid questionnaires being used for the analysis. The findings from this study were similar to those identified in Law et al., that students seem to be motivated by a set of intrinsic and extrinsic factors. Intrinsic factors being: setting difficult goals for themselves and having an interest in programming. Extrinsic factors being: social pressure and receiving rewards or recognition.

A quantitative study was conducted by Koulouri, Lauria, and Macredie (2014) with a number of different groups of students who were enrolled at a tertiary education institution. Each of the groups in the study had gone through different levels of learning aspects such as *problem-solving skills* and different types of programming courses were taught to some groups and not others. The researchers then measured each student in the different groups by giving the groups the same programming activity/assessment to measure their programming proficiency. It was found that students who had higher levels of experience in programming and students who had better problem-solving skills performed better in the programming activity/assessment.
The studies discussed in this section make strong reference to the variables that will be used to guide the current study. Each of the studies presented were conducted using a quantitative methodology with a survey instrument being used as the main form of data collection. The validity and significance levels that were achieved in these studies contributed towards a decision to opt for a quantitative methodology for the current study. While the planned study adopted a predominantly quantitative approach, an open-ended option has been included in the study's data collection instrument so that the benefits of a qualitative/mixed methods approach may be explored.

3.6 Study Site and Target Population

The site for the study was the Pietermaritzburg and Westville campuses of UKZN. However, due to the adoption of online learning the launch of the study's questionnaire was conducted during online lecture and practical sessions that were conducted via the MsTeams (3rd year, Honours and Master's) and Zoom (2nd year) video conferencing platforms.

3.6.1 Sample size

The estimated total population of the study was 420 university students from the IS&T Discipline. A census approach was adopted where the sample size chosen for the study was also the total population of the study which was 420 students. The research plan was to use the census approach and obtain a 100% response from the population because this was feasible and it would add immense value to the reliability of the study's outcomes. With the selected sample size, there was a high probability that the distribution of the sampling means would be approximately normal (Sekaran & Bougie, 2016). This created an opportunity to leverage the statistical power of parametric statistical analysis using a 95% confidence interval. These parametric statistical tests consist mainly of the one sample t-test that was used to ascertain whether the measures of central tendency such as the mean and median occured by chance or was statistically significant. Correlation analysis was performed by bivariate correlation testing (Pearson or Spearman). In cases of doubt, regarding the normality of the sampling distributions the back-up plan was to explore the possibility of using non-parametric statistical tests.

3.6.2 The Study's Sample

The population for the study comprised all Information System and Technology students currently studying a computer programming module. The population consists of 2nd and 3rd year Information Systems and Technology (IS&T) students including Honours and coursework Masters students. The purpose of using this set of students was that these students are appropriately positioned in an academic context as they were in the midst of undertaking a module that is focused on computer programming. The richness of the empirical phase of the study could have been enhanced by the participation of 1st year students. However, after careful review of the 1st year IS&T syllabus at UKZN, it was clear that the level of computer programming covered was not sufficient to make an accurate evaluation of students' computer programming abilities. The content at 1st year level is quite superficial and introductory. The complexities inherent in ensuring good academic performance in computer programming manifest when students engage with cognitively challenging computer programming constructs such as looping and conditional logic in conjunction with data structure and database manipulation. These assertions were based on material covered in the literature review section. In the case of 2nd and 3rd year students, there is an intensive engagement with all aspects of computer programming knowledge thereby providing the researcher with an ideal opportunity to obtain an insight into students' level of computer programming proficiency. The decision to include Honours and Masters students into the study's sample was based on the fact that these students would have been exposed to the rigours of computer programming tasks and they also engage with a module named Software Engineering that has a significant computer programming component.

3.7 The Data Collection

Data collection is the process that permits the researcher to collect, measure and analyse information in an established systematic way using standard validated techniques for a research study (Sullivan & Artino Jr, 2018). The main forms of data collection are surveys, interviews, observations, focus groups and document analysis (Mark Saunders, Lewis, & Thornhill, 2018). The current study used a survey approach that consisted of a questionnaire as the primary data collection instrument that was used to obtain a broad representation of computer programming knowledge and learning habits from the study's population. The questionnaire was launched during formal online lectures via the intervention of the academic staff members who were lecturing on the 2nd year, 3rd year and Honours and Master's lecturing programmes. Students were informed of the requirements of the questionnaire and were provided with an opportunity to complete the questionnaire during the practical sessions for the courses that contained computer programming content. The questionnaire was made available as an online survey via Google forms (accessible at: https://forms.gle/mtKtXc619XxGVhkC7). However, students who did not complete the questionnaire during the practical sessions were allowed to save the completed sections of the questionnaire and submit it at a later time of their convenience.

3.8 Construct Validity

This study employed a structured, survey questionnaire as the primary source of data for the study's empirical analysis. The questionnaire (Appendix A) was designed to resonate with the study's conceptual framework. According to Peter (1981) construct validity refers to the alignment between the constructs of the study (which are referred to as unobservable variables specified at a conceptual level) and the questionnaire items that are used to obtain a tangible measure of that construct. There is no precise measure of construct validity. However, it can be inferred through techniques such as exploratory and confirmatory factor analysis. The strategy is to ascertain whether the questionnaire items contributed by explaining/accounting for the variance in the dependent variable. A viable strategy to ensure construct validity is to align questionnaire items to previous studies where these constructs and items have been validated.

3.8.1 Validation of the Study's Questionnaire Items

The study's main constructs were subjected to theoretical validation by using pervious research efforts with a similar objective as the current and also included constructs that were been identified in the study's conceptual framework.

The discussion on construct validity is classified according to the study's main constructs:

- *Self-Efficacy (SE)*

Bandura (2006) developed a guide for constructing SE scales. This guide, which has reached seminal status, has been used extensively by researchers to measure this concept. Ramalingam and Wiedenbeck (1998) used Classical Measurement Theory and factor analysis techniques to confirm the validity of 30 questionnaire items linked to the Bandura scale to measure self-efficacy in the domain of computer programming. Askar and Davenport (2009) adapted the Bandura scale to measure SE of students during a Java programming course delivered to Engineering students at a university. The current study was guided by the afore-mentioned studies and 12 questionnaire items were adapted from these studies to align with the context of the current study

- *Intrinsic and Extrinsic Motivation*

According to Nielsen (2018) the Motivational Strategies for Learning Questionnaire (MSLQ) is a widely used instrument to measure intrinsic and extrinsic motivation levels for students in higher education. Nielsen used this questionnaire as a starting point to measure the levels of intrinsic motivation (IM) and extrinsic motivation (EM) of

Psychology students to their academic studies. The outcome from this study was a set of validated questionnaire items to measure IM (4 items) and EM (4 items). In a s similar study Amabile, Hill, Hennessey, and Tighe (1994) used the Work Preference Inventory (WPI) over a period of 8 years to collect data on motivational traits of college students and working adults. The results from their study indicate the WPI is a reliable indicator of IM and EM in various contexts. An examination of the MSLQ and the WPI instruments to measure IM and EM revealed substantial overlap of the questionnaire items. A set of 7 questionnaire items (4 for IM and 3 for EM) was identified as applicable for the context of the current study.

- *Learning Styles*

As discussed in the literature section of the study, learning styles is an attribute that manifests in the form of deep and surface learning. In terms of quantifying this abstract phenomenon, Mahatanankoon and Wolf (2021) provided a set of validated questionnaire items (informed by seminal work on this topic by Marton and Säljö (1976)) to measure deep and surface learning traits exhibited by students in the context of computer programming. In the context of the current study 6 questionnaire items were adapted (3 for deep learning and 3 for surface learning) for the current study.

- *Problem Solving Ability*

As discussed in the study's literature review, problem solving ability was operationalized for the purpose of computer programming inquiry by aligning this ability to the students' mental model visualisation of a computer programming task to a formal programming language notation was used to represent a solution to the task. To measure this ability, the study was guided conceptually by the pragmatics of computer programming (Du Boulay, 1986) that allude to the challenges of solving computer problems by dealing with the syntax and semantics of a computer programming language. This strategy was extended operationally by leveraging questionnaire items to measure students' problem solving ability suggested by Tukiainen and Mönkkönen (2002) and Koulouri et al. (2014). A series of 10 questionnaire items were used that targeted student's cognitive ability in developing a mental model of the problem situation and identifying a correct solution to the problem. These questions were devised to ascertain student's ability to be cognitively adept at prediction, conditional and iteration logic as well as data structure comprehension. In the instances where computer programming code was required to contextualise the question, the C# programming language was used because it is the current language of computer programming instruction used in the IS&T Discipline at UKZN. The relevance of these constructs as a reliable indicator of problem-solving ability were confirmed during the pilot study where academics who are experienced in the lecturing and research of computer programming were consulted.

Table 3.1: Summary of the Questionnaire Validity

Construct	Reference (adaptation)	Number of Questionnaire Items
Self-Efficacy	Askar and Davenport (2009)	12 items
Learning Styles	Mahatanankoon and Wolf (2021)	6 items
Motivation	Amabile et al. (1994)	7 items
Problem Solving Ability	Tukiainen and Mönkkönen (2002)	10 items

3.8.2 Data Reliability

The routine check for data reliability was conducted via the Cronbach Alpha test. As is suggested in Sekaran and Bougie (2010), a Cronbach Alpha value that is greater than 0.7 is indicative of very good data quality. At this stage, as Peter (1981) suggests the exercise in establishing construct validity is only tentative and is ideally achieved through a process of iteration where the empirical data collected in the study is subjected to discriminant validity tests that are guided by Cronbach's alpha and factor analysis. Irrespective of the outcome of the empirical validation tests, the theoretical basis for the study's conceptual model should always take precedence.

3.9 Questionnaire Design

The questionnaire (see Appendix A) was classified into 4 sections summarised in Table 3.2.

Table 3.2: Sections in the Questionnaire

Section Label	Purpose	Number of Components
Section A	Demographic- Part 3 of Section A contains reference to an estimation by the respondent of their average computer programming level of achievement (a major dependent variable in the study).	3 parts
Section B	Behavioural, cognitive and problem-solving questionnaire	3 parts

	items (the major independent variables of the study)	
Section C	Open-ended response	1 part

3.10 Pilot Study and Survey Protocol

The questionnaire was discussed with academics involved in the teaching of computer programming in the Information Systems and Technology (IS&T) at UKZN. Comments and suggestions were incorporated into the questionnaire so that it represented a cogent, logical document that flowed from one construct to the next and aligned with the programming experiences of students in the IS&T Discipline at UKZN. The questionnaire was piloted with 2 IS&T Master's students and 2 students from the IS&T Honours class. The input received during the pilot studies was that the questionnaire was too long and a few questions were ambiguous. Also, the original questionnaire required students to solve computer programming related problems to ascertain their problem-solving ability. While these questions were not too intensive, the response from the pilot study was that these questions should be phrased as multiple-choice questions. All suggestions made during the pilot study deliberations were considered and in conjunction with the supervisor of the current study, many of these suggestions were implemented.

3.11 Ethical Consideration

The researcher applied for ethical clearance from the Research Office at UKZN. A gatekeeper application was made to the registrar's office to obtain permission to collect data within the UKZN campus. Both the ethical clearance and gatekeeper applications were successfully granted. In terms of the survey protocol, the study's respondents were informed of their voluntary participation in the study and in compliance with the Personal Protection of Information (POPI) Act, no personal information was collected that could be used to directly identify the study's respondents.

3.12 Planned Data Analysis

The data analysis that was planned was descriptive and inferential statistical analysis. The descriptive statistics consisted of frequencies, mean, median and standard deviation statistics. The descriptive results are displayed by stacked bar graphs and histograms. These data visualisation techniques were used to provide an overall view of the empirical evidence with regards to the study's main constructs such as previous experience, problem solving ability, self-efficacy and performance in a formal computer programming assessment. The inferential statistics consists mainly of the one sample t-test and tests of normality. The correlation between the main constructs of the study were analysed by making use of the Pearson Correlation Co-efficient or the Spearman rho, bivariate and multiple regression and path analysis. The study also leveraged Confirmatory Factor Analysis techniques to ensure discriminant validity and an alignment with the study's conceptual framework. The dependent variable, which is the performance in computer programming assessment, was determined by students' responses where they will be asked to provide an approximate value that quantifies their performance (Section A, Part 3 of the questionnaire). A correlation analysis (Spearman or Pearson) was conducted between students' survey-based responses to the problem-solving tasks administered in the survey (Questionnaire, Section B, Part 3) and their approximation of their performance in computer programming assessment.

3.13 Summary of the Research Methodology

In conclusion the study has been set up to leverage the conceptual framework, engage in data collection via the study's research instrument and proceed to the data analysis section. The alignment between the study's conceptual framework and the data collection instrument has been presented in the current chapter. A theoretical explanation and presentation of the construct validity of the questionnaire items have been presented. The empirical validity of these items together with the predictive capacity of the study's conceptual model is presented in Chapter 4.

CHAPTER FOUR – Data Analysis

4.1 Introduction

Chapter 4 presents a statistical summary of the responses received during data collection. The study's data was obtained online, via Google Forms and downloaded into a spreadsheet format. While a few of the illustrations were developed using the MsExcel spreadsheet application, the majority of the analyses, interpretations and visualisations were generated via SPSS version 28.

The primary data collection instrument consisted of a questionnaire that comprised of 2 main sections (See Appendix A). The 1st section labelled as Section A consisted of demographic questions and questions pertaining to levels of experience in the domain of computer programming and systems analysis and design (SAD). The 2nd section of the questionnaire comprised of the core aspects that addressed the main objectives of the study. This design view of the 2nd section of the questionnaire is provided in Table 4.1.

Table 4.1: Section Two of the Questionnaire

Section label	Type of response	Concept	Number of Items
Part One	Likert Scale	Intrinsic and Extrinsic motivation (pertaining to academic performance in computer programming)	7
Part Two	Likert Scale	Learning styles	6
Part Three	Likert Scale	Self-Efficacy	12
Part Four	MCQ	Problem Solving and Computing Mental Model	10
Part Five	Open ended text	Open ended response on student experience/comments pertaining to academic performance in computer programming	1

The questions for Part One, Part Two and Part Three of the questionnaire were presented to the respondents as a 5-point Likert Scale type questions. The questions were positively worded and coded into SPSS using a strategy of 5=strongly agree and 1=strongly disagree. In Part Four of the questionnaire, the respondents were presented with a series of questions pertaining to the ability to leverage the cognitive domain and demonstrate knowledge/comprehension of conditional, logical and data structure-oriented problems. The questions in this section of the questionnaire were structured as multiple-choice questions (MCQs). In Part Five of the questionnaire, the respondents were provided with an open-ended section where they could provide a response regarding their experience of learning computer programming and making suggestions on how this learning could be enhanced.

The current chapter has been designed according to the following plan:

- A presentation of the demographic data using tables and visualisations; this section contains frequencies and bar graphs to provide an overall view of the data. The main data items presented were demographics and background information of the participants. Descriptive analysis was carried out and presented using bar charts and pie graphs
- Inferential statistics are presented by ensuring data reliability (Cronbach Alpha), data compliance in terms of testing for normality and the use of t-tests to establish the significance of the mean value that represents each construct; Confirmatory factor analysis (CFA) was conducted on the study's conceptual model by inputting the study's current dataset into the CFA model; relationships between variables were explored via bivariate correlations analysis to establish whether there were significant correlations and to identify the strength and direction of significant correlations; multiple regression analysis to ascertain whether the correlations can be expressed in a predictive linear relationship; and path analysis to find an optimal fitting model that explains maximum variance in the dependent variable.
- Thematic analysis was also carried out on Section C, where the respondents were provided with an opportunity to openly express their thoughts on academic performance in computer programming.
- The results were analysed with predominant quantitative techniques; however, the open-ended data represented an opportunity to triangulate the study's quantitative data with the qualitative responses and provide a more enriched discussion of results.

4.2 Sample and Response Rate

The data collection strategy was launched as a census sampling effort where all students who are registered for Information Systems and Technology (IS&T) courses were requested to respond to the questionnaire. The questionnaire for the study was made available via Google Forms and a link to the form was posted on the Learning Management System (LMS) used at UKZN for 2nd year, 3rd year, Honours and coursework Master's students. Academic staff members also made announcements to the students to respond to the questionnaire and in the case of the IS&T 3rd year and Honours classes, students were allowed to complete the questionnaire as a practical task (that was classified as optional). Students were allocated time during their 3rd year, Honours and Master's classes to complete the questionnaire including the set of problem-solving questions. However, students were given the latitude of using more time than what was available for the official lecture to complete the full questionnaire and submit later than the expected time. Also, the cohort of responses included responses from students who were not available during the official dissemination of the questionnaire.

The population for the study was identified on the basis that all of these students would have had experience in computer programming at the level that was required in this study. The study's population was identified as 420. Only 133 valid responses were received. This represents a response rate of 32%.

4.3 Demographic and Background Information of Participants

Section A of the questionnaire was designed predominantly to obtain demographic and background information from the study's respondents. The strategy adopted here was to use an expansive approach that provided with an opportunity to scan the demographic data with the possibility of identifying any significant relationships that may emerge with the study's main constructs. However, Section A did make specific reference to previous programming experience which does have a tangible link to the study's main constructs. The demographic data pertaining to the level of study is presented in Figure 4.1.

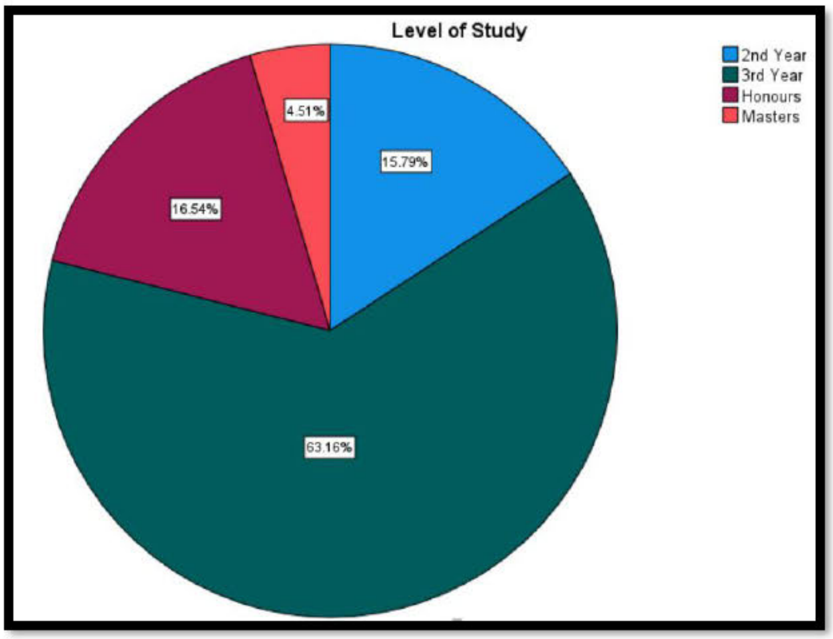


Figure 4.1: Graphical illustration of Level of Study

The academic college of affiliation of the study's respondents has been recorded and this data may present an opportunity for further analysis. As can be observed from Figure 4.2, 64% of the respondents are from the College of Law and Management Studies (LMS) and 36% are from the College of Agriculture, Engineering and Science (AES).

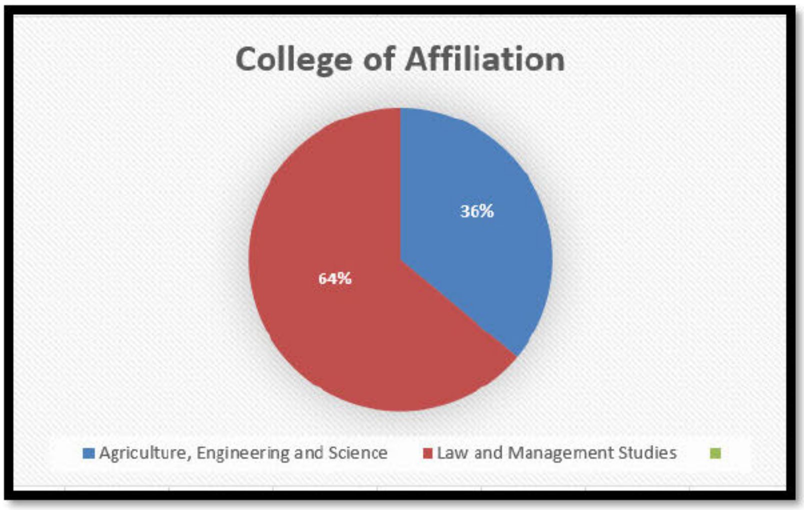


Figure 4.2: Graphical illustration of College of Affiliation

A summarised view showing an approximation of the years of computer programming experience acquired by the study's respondents is provided in Figure 4.3.

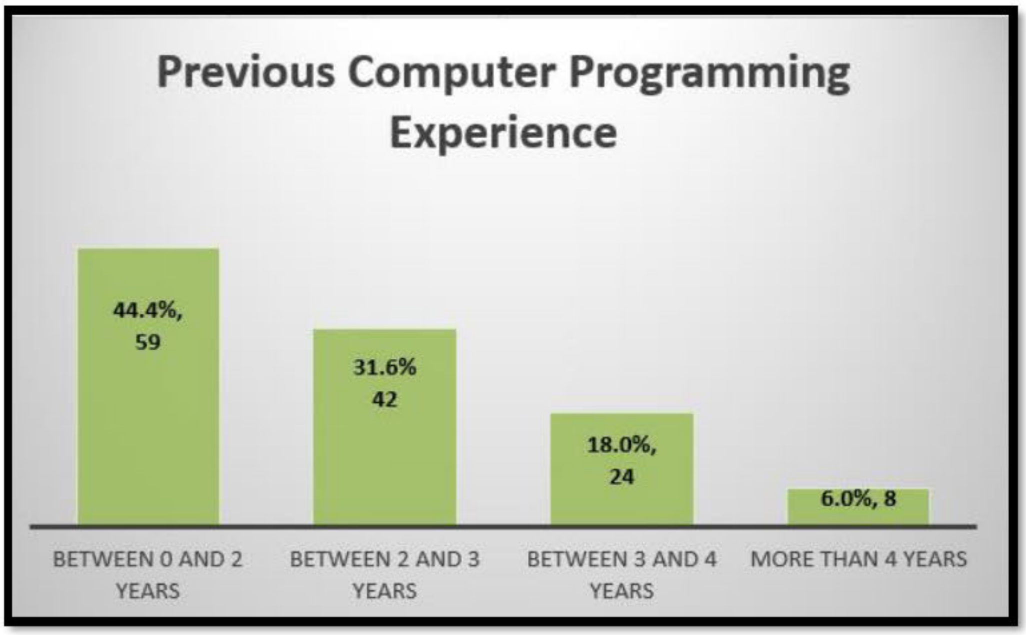
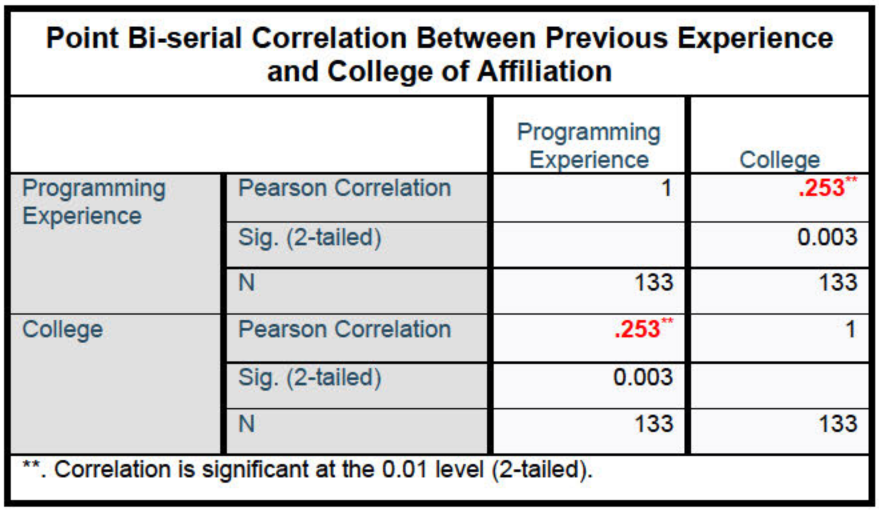


Figure 4.3: Previous Computer Programming Experience

The significance of the information presented in Figures 4.2 and 4.3 is that the majority of the study's respondents are affiliated to LMS and the researcher is aware that based on curriculum specifications, students from AES will in all probability have greater previous experience of computer programming engagement. This brings the construct of previous experience into contention and at this stage, the researcher was "interested" to see what the correlation between these variables turned out to be. This knowledge will assist in guiding further statistical testing and provide convergence towards a discussion of the study's research questions. A point biserial correlation was conducted between the respondents' college of affiliation and the number of years of previous computer programming experience. The dichotomous variable "College of Affiliation" was recoded in SPSS with LMS being assigned a value of 1 and AES being assigned a value of 2. This correlation was generated to obtain a Pearson r value as well as a Spearman rho value. In both instances, the correlation was a weak, but positive correlation (r=0.25, n=133, p=0.03 and rho =0.25, n=133, and p=0.03). The Pearson correlation analysis between computer programming experience and the college of affiliation is shown in Table 4.2 The interpretation that is coupled with these results is that students affiliated to AES have a higher correlation with greater levels of computer programming experience.

Table 4.2 Correlation Between College of Affiliation and Previous Experience



Point Bi-serial Correlation Between Previous Experience and College of Affiliation

		Programming Experience	College
Programming Experience	Pearson Correlation	1	.253**
	Sig. (2-tailed)		0.003
	N	133	133
College	Pearson Correlation	.253**	1
	Sig. (2-tailed)	0.003	
	N	133	133

**. Correlation is significant at the 0.01 level (2-tailed).

4.4 Reliability Testing

Gliem and Gliem (2003) suggests that when Likert scales are used in a study, the reliability of the data collection instrument needs to be established. According to Bujang, Omar, and Baharum (2018), Cronbach's alpha provides a very good measure of the inter-item reliability in a questionnaire. The values for Cronbach's alpha ranges from 0 to 1 with the higher values indicating that the questionnaire items contribute towards measurement of the construct under inquiry while lower values indicate that some or all of the items are not making a contribution towards measuring the value of a specific construct in the study. For the current study, there were 3 constructs that were measured using a Likert-scale type of response. The outcome of the Cronbach alpha reliability tests that were conducted on these 3 constructs are presented in Table 4.3

Table 4.3 Cronbach alpha analysis

Construct	No of Likert Scale Items	Cronbach's alpha
Motivation (Intrinsic/extrinsic)	7 (abbreviated as IM1 to IM7)	0.64
Learning Styles	6 (abbreviated as LS1 to LS6)	0.57
Self-Efficacy	12 (abbreviated as SE 1 to SE12)	0.91

Sekaran and Bougie (2016) provide guidance on the use of Cronbach alpha by suggesting that a co-efficient value that is less than 0.6 is "considered to be poor" (p.311) while those in the range of 0.7 and above are considered to be good. In Table 4.2, it can be observed that the Cronbach alpha value for Learning Styles is 0.57 and according to Sekaran and

Bougie (2016), the questionnaire items used to measure this construct is not ideally reliable. Gliem and Gliem (2003) do however advise that if the Cronbach alpha value achieves a value in the range between 0 and 3, then all items measuring a construct do contribute consistently to the overall measurement achieved. Upon further inquiry it was found that for the construct of Learning Styles, if Question item 3 ("I test myself on important topics until I understand them completely") is removed, then the Cronbach alpha value for the Learning Styles construct increases to a value of 0.61. The improved Cronbach Alpha value creates a temptation to remove Question item 3 from the set of 6 question items that measure learning styles. However, Youngman (1979) advises that while it is desirable to obtain high Cronbach alpha values, the internal consistency of questionnaire items does not have to be perfect. At this stage, the researcher has opted to include Question item 3 and subject the constructs to further validity testing in the form of factor analysis that is conducted in the section that follows.

4.5 Factor Analysis

Sekaran and Bougie (2016) make a specific reference to factor analysis as a valid statistical technique that could be used to reduce the complexity of a conceptual model by eliminating variables that are not tightly coupled with the latent variables or the main underlying constructs from the theoretical model. The process of variable reduction is conducted under the theory that, if the conceptual model does not have an alignment with the study's data, then the conceptual model needs to be re-arranged or re-configured so that it has an optimal alignment with the conceptual model. According to Thu, Dang, Le, and Le (2021), this process of fitting the conceptual model to the study's data is referred to as confirmatory factor analysis (CFA) which is a crucial process in ensuring construct validity.

4.5.1 Confirmatory Factor Analysis

The CFA conducted for the study's data was confined to those constructs that consisted of multiple "observed" variables. These were Motivation, learning styles and self-efficacy and each of these constructs were measured using a Likert-scale as indicated in Table 4.3. According to Mark Saunders et al. (2018) factor analysis entails the construction of a correlation matrix between a study's variables. The study's latent variable set consisted of 25 observed variables. These variables were identified from the study's literature review and conceptual model.

The knowledge that the empirical component of the study is informed by a conceptual model creates an opportunity for the use of CFA. According to Hurley et al. (1997) and Goodwin (1999), once a study has *a priori* model CFA may be used as a variable reduction technique so that there is optimal factor loadings between the study's observable variables and the latent variables (main constructs). The CFA technique was used in the current study to derive various specification models that will have an ideal fit with the study's data.

The objective to obtain an ideal conceptual model fit for the study's data is driven by the need to ensure that the data analysis converges to a point where the study's research questions can be answered with a measure of confidence. Based on various permutations of variable arrangements that were produced by the factor analysis exercise, a viable model that has been settled upon is presented in Figure 4.4. One of the objectives of construct validity is that the main constructs should have a high correlation with the study's variables (Farrell & Rudd, 2009). As can be observed from Figure 4.3, all the correlations are within an acceptable range (>.5) and these correlations have been observed to be significant ($p<0.05$) which is indicative of a "good fit" between the study's observed variables and the latent variables. However, in order to achieve this "good fit" the number of variables has been reduced from 25 to 17 (refer to Table 4.2). While this reduction in the number of variables will ensure a cogent model that may be used to ensure greater significance levels during further correlation analysis, the study's aggregate data (mean, median and standard deviation) will be computed with the original variable set. This will provide the researcher with a more extensive set of resources to enable a "richer" discussion of the results obtained from the study's data analysis.

In terms of the overall fit between the 3 constructs illustrated in Figure 4.3, the following parameters were observed:

- The Comparative Fit Index (CFI) index measurement should be closer to 1 (Stapleton, 1997)
- The Tucker Lewis Index (TLI) should be in the range of 0,9 to 1 (Thu et al., 2021)
- The root mean square error of approximation (RMSEA) should be less than 0.08 (Thu et al., 2021).

In the context of the current study, the "model fit" indicators arising out of Figure 4.4 are CFI=0.91, the TLI =0.9 and the RMSEA =0.082. These results indicate that the empirical model that will be used for the data analysis for the study is not a perfect fit to the study's data but it does have a close alignment with the suggested test statistics to guide knowledge on the level of alignment to ensure a “good fitting” model.

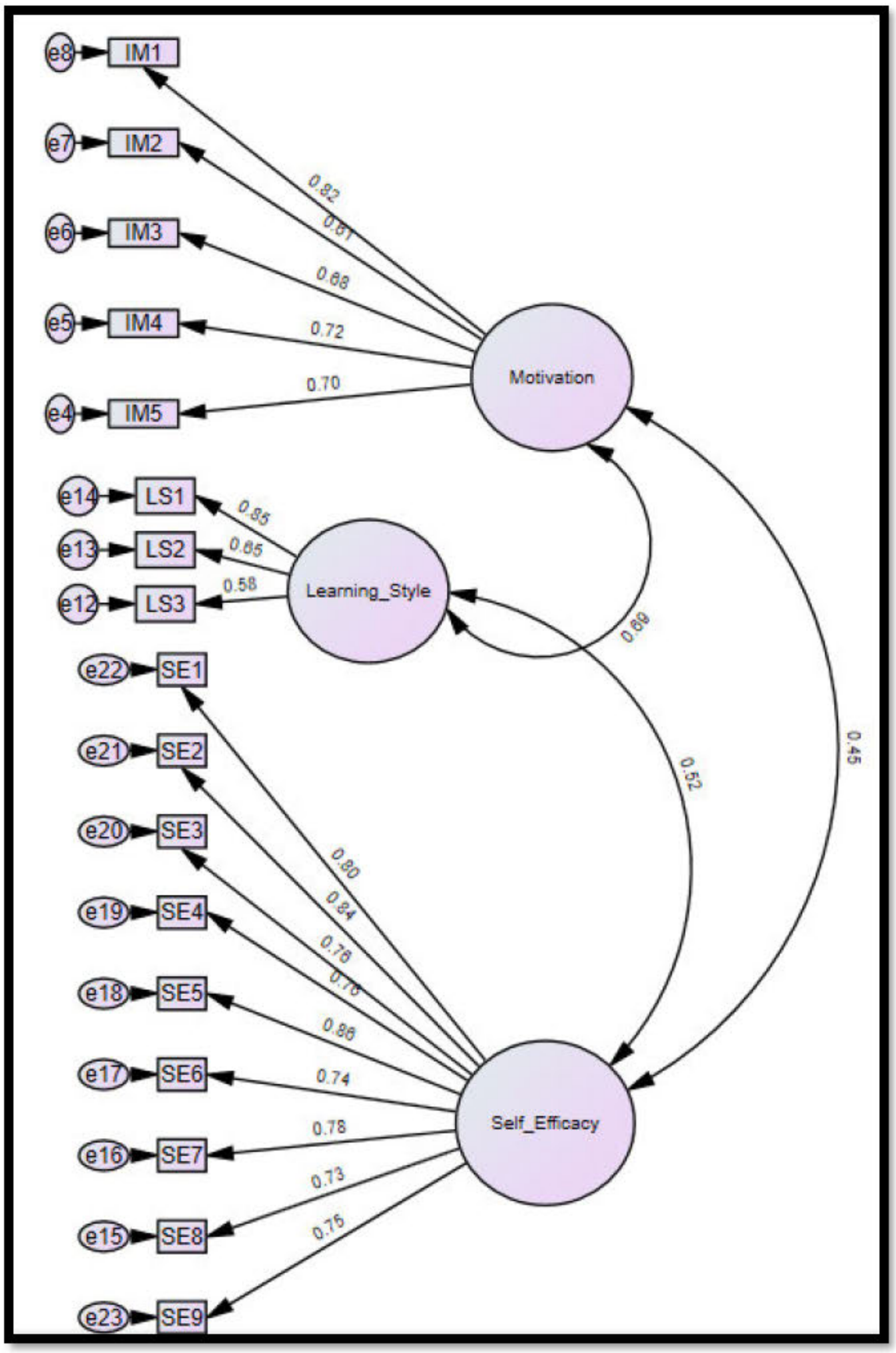


Figure 4.4: Confirmatory Factor Analysis of the Observed Variables

From an item reliability perspective, the improvement in the internal consistency of the empirical model is confirmed by the reworked Cronbach alpha values shown in Table 4.3

Table 4.4 Re-worked Cronbach alpha analysis Values

Construct	No of Likert Scale Items	Cronbach's alpha
Motivation (Intrinsic/extrinsic)	5 (abbreviated as IM1 to IM5)	0.83
Learning Styles	3 (abbreviated as LS1 to LS3)	0.72
Self-Efficacy	9 (abbreviated as SE 1 to SE9)	0.93

4.6 Inferential Analysis

Saunders et al. (2009) provides guidance on the process of extrapolating results from a study's sample to the study's population, referred to as statistical inference. This enables a researcher to determine the probability that the results obtained from the sample is purely by chance or has a high probability of inference to the study's population.

The plan going forward is to subject the study's data (individual and aggregated) to the following tests as suggested in Sekaran and Bougie (2016)

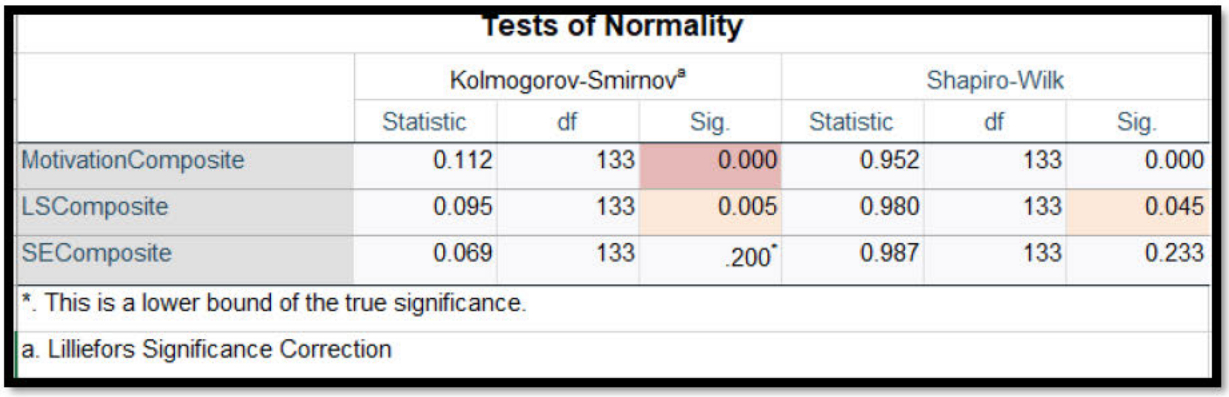
- Test of Normality – to provide guidance on the use of parametric or non-parametric testing or a mix of both
- A presentation of the study's frequency distributions
- The invocation of t-tests – to ensure that the aggregated values or measures of central tendency reported from the Likert scale data representing the study's main constructs is significant
- The invocation of bivariate and multi-variate correlation analysis that is guided by the study's conceptual model and the outcome of the CFA exercise
- The invocation of path analysis and structural equation modelling to examine the correlations between the study's main variables and the explore the opportunity of developing a better correlational model between the study's variables

4.6.1 Tests for Normality

The tests for normality somewhat controversial in the annals of statistical theory (Norman, 2010). In its purest form, the suggestion is that if the means of the sampling distribution is normal, then the inferential statistics should be conducted using parametric statistical tests and if it is non-normal, then non-parametric tests should be invoked (Saunders et al., 2009). However, in a seminal article by Norman (2010), the preceding rules regarding parametric and non-parametric testing is disputed and the suggestions made is that the error between the 2 types of testing is usually minimal and since parametric tests are more robust, then if the sample size is in excess of 30 and guided by the Central Limit theorem, parametric testing will be the better option (Hoskin, 2012; Islam, 2018). However, during the course of the data analysis presented for the current study, both parametric and non-parametric testing options will be explored.

The study's main constructs where subjected to the Shapiro-Wilk (SW) and Kolmogorov-Smirnov (KS) test of normality and presented in Table 4.5

Table 4.5 Tests for Normality



Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
MotivationComposite	0.112	133	0.000	0.952	133	0.000
LSComposite	0.095	133	0.005	0.980	133	0.045
SEComposite	0.069	133	.200*	0.987	133	0.233

*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction

When it comes to normality testing, the null hypothesis states that the sampling distributions are NOT normal. As can be observed in Table 4.4 the constructs of Motivation and Learning Styles (LS) pass the test for normality (null hypothesis rejected, $p<0.05$) because the probability that the sampling distribution is not normal is quite low (p=0.00 and p=0.045 respectively). However, the construct of Self-efficacy (SE) fails the test of normality (null hypothesis accepted, $p>0.05$) because the probability that the sampling distribution in not normal is quite high. It is also worth taking note of the observation that Learning styles passes the test for normality only marginally (p=0.045). Given the closeness of these tests to accept or reject the condition of normality, the strategy adopted for the subsequent analysis will entail a mix of parametric and non-parametric tests.

4.6.2 Frequency Distributions for the Construct of Motivation

From an overview perspective, the construct of Motivation (to learn computer programming) has been represented by 8 questionnaire items where 5 represent intrinsic motivation (IM) and 3 represent extrinsic motivation (EM). An overall presentation of the responses is provided in Figure 4.5.

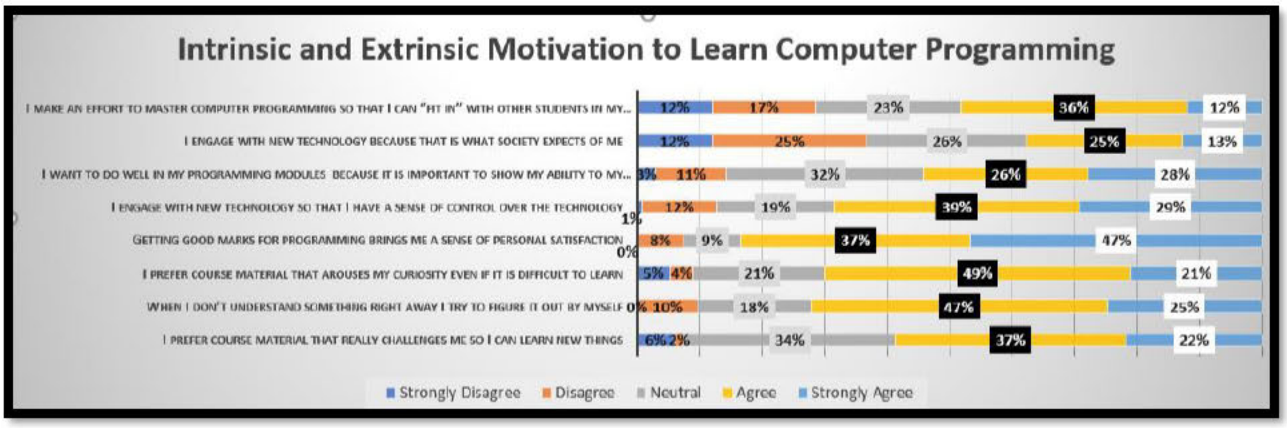
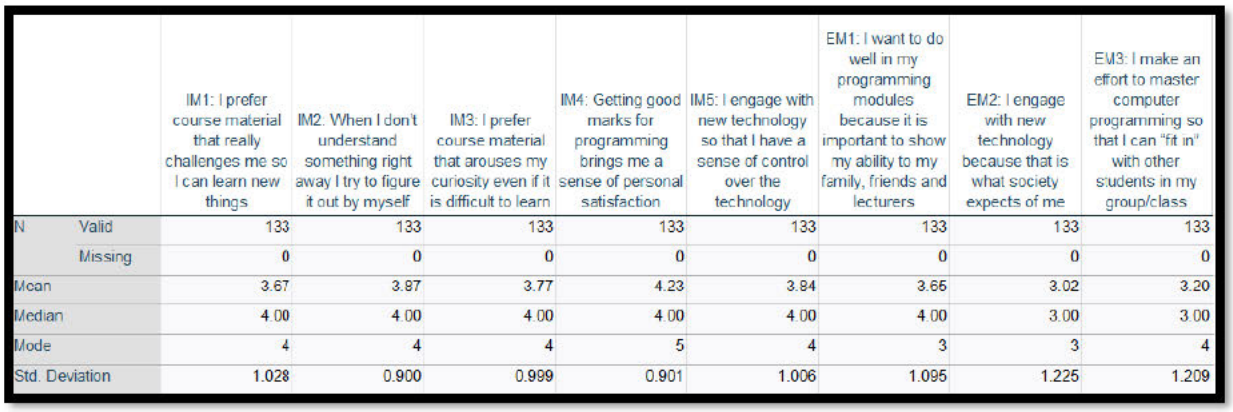


Figure 4.5: Overall view of responses for the construct of Motivation

The questions were phrased positively towards higher levels of intrinsic and extrinsic motivation. The aggregated outcome of the frequency representation for the Motivation construct is evidenced by the results shown in Table 4.6

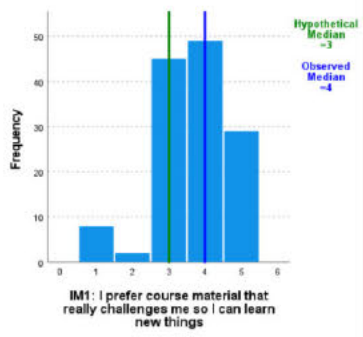
Table 4.6 Measures of Central Tendency for Motivation

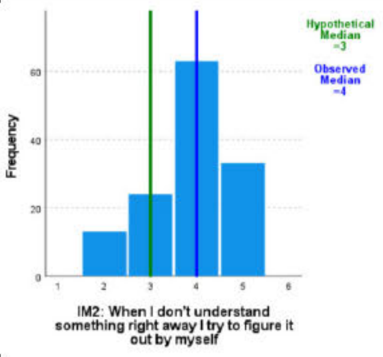


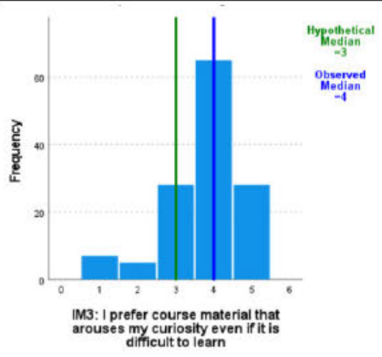
		IM1: I prefer course material that really challenges me so I can learn new things	IM2. When I don't understand something right away I try to figure it out by myself	IM3: I prefer course material that arouses my curiosity even if it is difficult to learn	IM4: Getting good marks for programming brings me a sense of personal satisfaction	IM5: I engage with new technology so that I have a sense of control over the technology	EM1: I want to do well in my programming modules because it is important to show my ability to my family, friends and lecturers	EM2: I engage with new technology because that is what society expects of me	EM3: I make an effort to master computer programming so that I can "fit in" with other students in my group/class
N	Valid	133	133	133	133	133	133	133	133
	Missing	0	0	0	0	0	0	0	0
Mean		3.67	3.87	3.77	4.23	3.84	3.66	3.02	3.20
Median		4.00	4.00	4.00	4.00	4.00	4.00	3.00	3.00
Mode		4	4	4	5	4	3	3	4
Std. Deviation		1.028	0.900	0.999	0.901	1.006	1.095	1.225	1.209

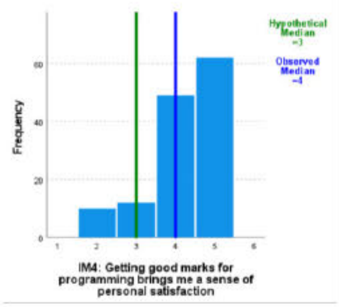
As can be observed in Table 4.6 the mean response is in excess of 3 (M>3) and the median is greater than or equal to 3 in all cases (Mdn>=3). To establish whether the mean and median values are significant measures of central tendency for the dataset shown in Table 4.5 or whether these values occur by chance, the one sample t-test is used. The decision to use the t-test is guided by suggestions in Saunders et al. (2009) where it is claimed that the assumption of normality could be violated without much consequence. According to DeCoster and Claypool (2004), the one sample t-test may be used to determine if the mean of a sample is significantly different from a hypothesised value. In the context of the current data (Table 4.5), the null hypothesis is that the mean (parametric) is equal to a hypothesised neutral value of 3 (H_0:M=3) and median (non-parametric) is equal to a hypothesised neutral value of 3 (H_0:Mdn=3). In both cases the alternate hypothesis is that these measures of central tendency are significantly different from 3 ($H_a \neq 3$). The t-test to establish the significance of the measures of central tendency have revealed results that are identical to the non-parametric equivalent test, which is the one-sampled Wilcoxon signed rank test illustrated in Table 4.7.

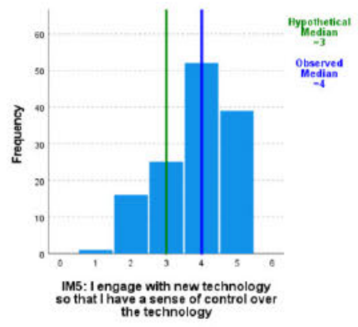
Table 4.7 One Sampled Wilcoxon Signed Rank Test for Motivation

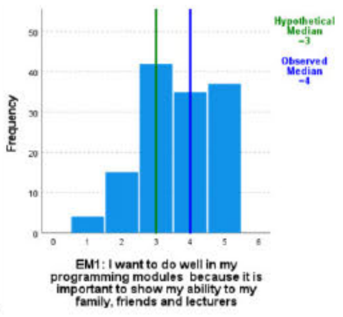


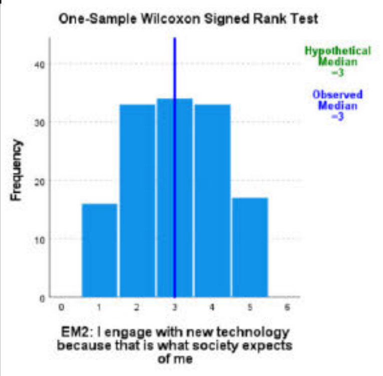


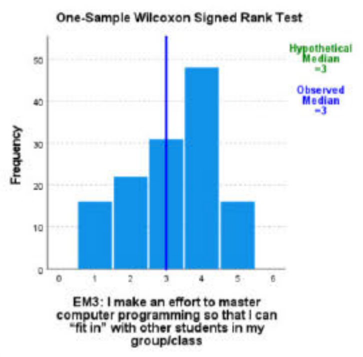












As can be observed in Table 4.7 the observed medians were significantly greater than the hypothesised median of 3 in 6 of the 8 (75%) questionnaire items. Five of the 6 items were aligned to observable measures of intrinsic motivation. The implication from this analysis is that there is a significant ($p<0.05$) tendency by the respondents to opt for responses that indicate high levels of intrinsic motivation to learn computer programming. This result indicates that the majority of the study's respondents have a positive disposition to intrinsic motivation factors that guide them towards achieving a good learning of computer programming. This conclusion cannot be made when it comes to the extrinsic motivation factors which were items 7 and 8 on the questionnaire did not yield a significant ($p>0.05$) outcome, thereby reducing the prospect of a 95% confidence with conclusions made in terms of extrinsic motivation.

4.6.3 Frequency Distributions for the Construct of Learning Styles

From an overview perspective, the construct of Learning Styles has been represented by 6 questionnaire items where 3 of the items were positively worded in favour of a deep learning style and the remaining 3 items were positively worded in favour of a surface learning style. Essentially the learning style component bears testimony to student's learning behaviour patterns when it comes to obtaining a deep understanding of computer programming or a surface understanding that enables them to pass computer programming assessment. An overall presentation of the responses is provided in Figure 4.6.

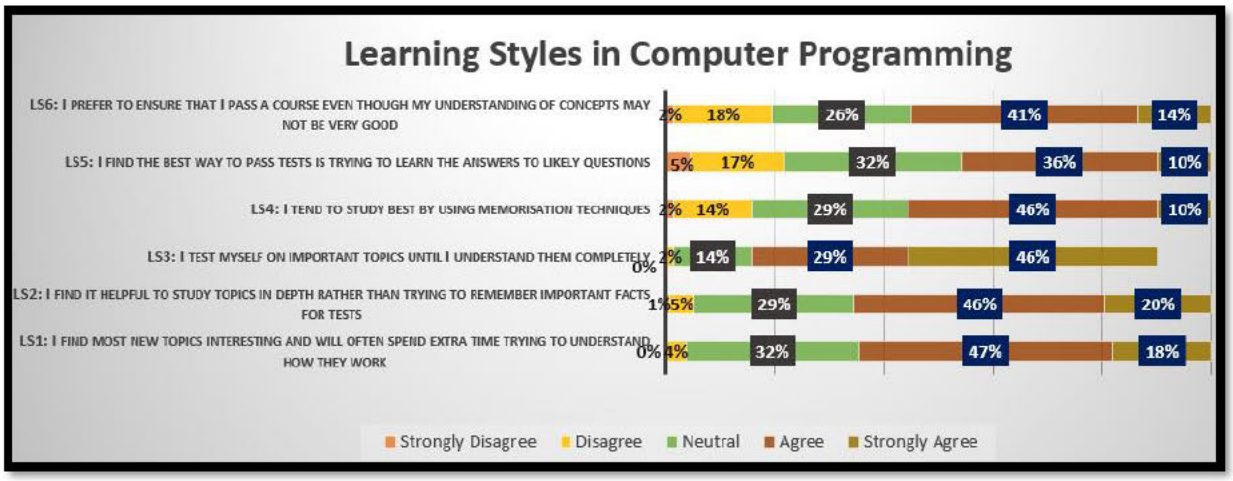
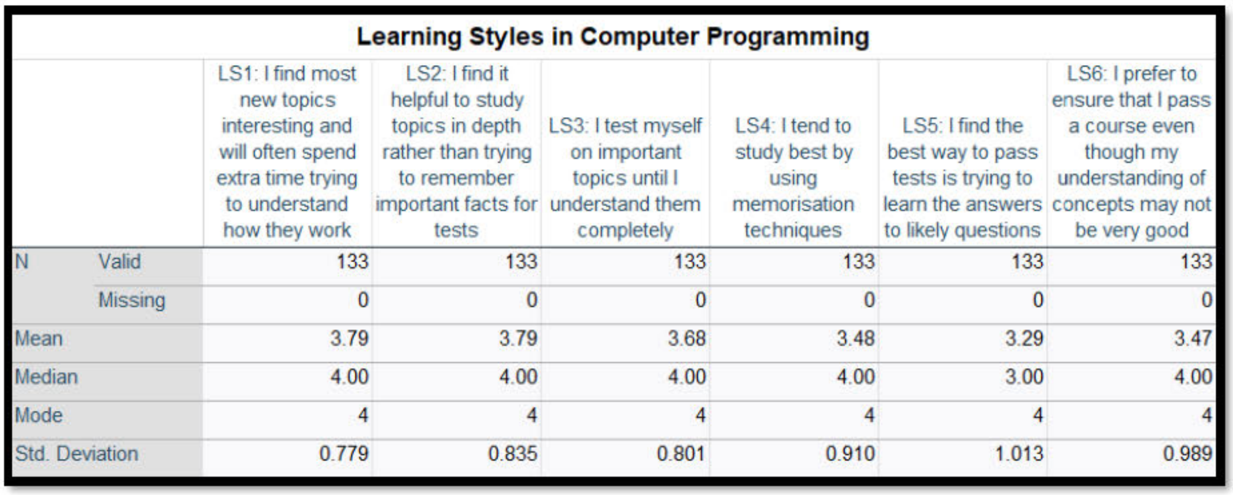


Figure 4.6: Overall view of responses for the construct of Learning Styles

The questions were phrased positively towards the adoption of higher levels of a "deep learning" style towards computer programming. The aggregated outcome of the frequency representation for the Learning Styles construct is evidenced by the results shown in Table 4.8

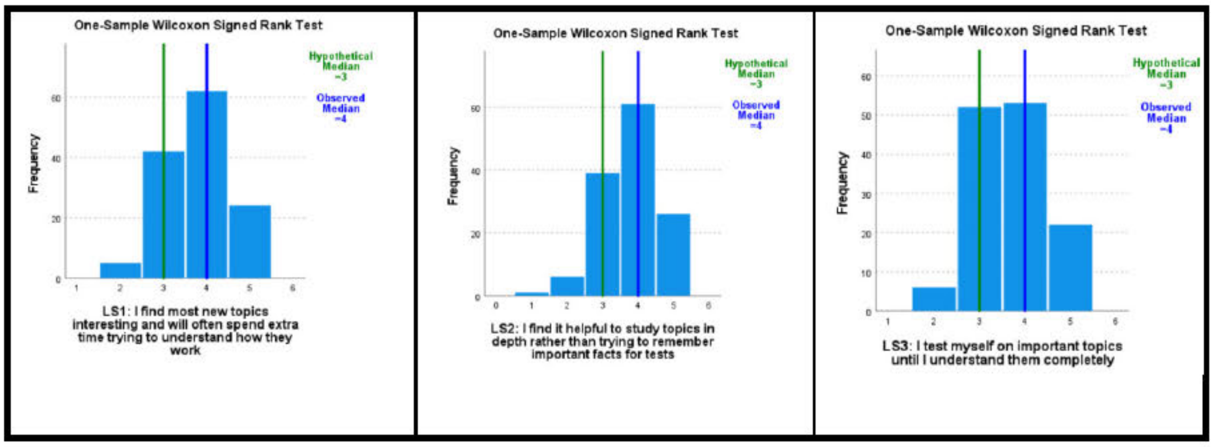
Table 4.8 Measures of Central Tendency for Learning Styles

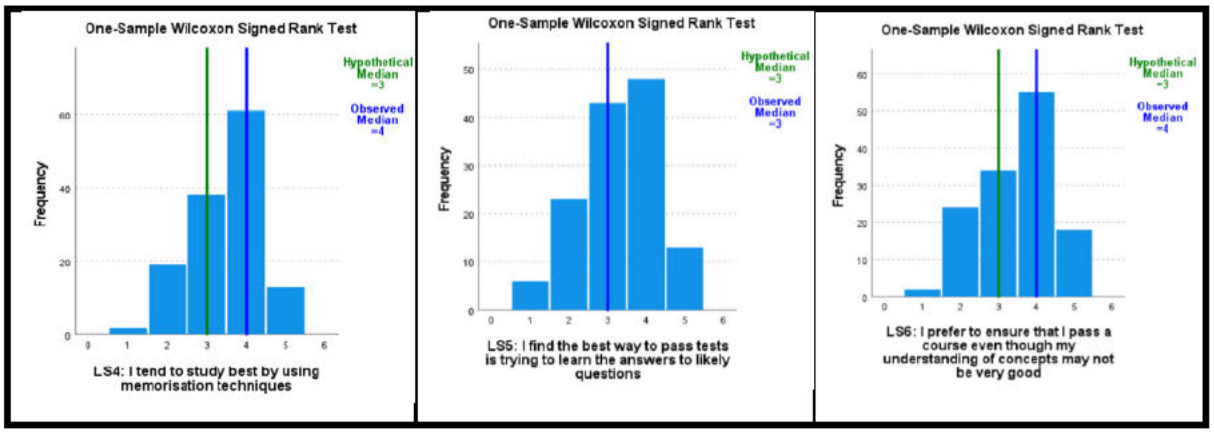


Learning Styles in Computer Programming		LS1: I find most new topics interesting and will often spend extra time trying to understand how they work	LS2: I find it helpful to study topics in depth rather than trying to remember important facts for tests	LS3: I test myself on important topics until I understand them completely	LS4: I tend to study best by using memorisation techniques	LS5: I find the best way to pass tests is trying to learn the answers to likely questions	LS6: I prefer to ensure that I pass a course even though my understanding of concepts may not be very good
N	Valid	133	133	133	133	133	133
	Missing	0	0	0	0	0	0
Mean		3.79	3.79	3.68	3.48	3.29	3.47
Median		4.00	4.00	4.00	4.00	3.00	4.00
Mode		4	4	4	4	4	4
Std. Deviation		0.779	0.835	0.801	0.910	1.013	0.989

As can be observed in Table 4.8 the mean response is in excess of 3 (M>3) and the median is greater than or equal to 3 in all cases (Mdn>=3). To establish whether the mean and median values are significant measures of central tendency for the dataset shown in Table 4.8 or whether these values occur by chance, the one sample t-test is used. In the context of the current data (Table 4.7), the null hypothesis is that the mean (parametric) is equal to a hypothesised neutral value of 3 (H_0:M=3) and median (non-parametric) is equal to a hypothesised neutral value of 3 (H_0:Mdn=3). In both cases the alternate hypothesis is that these measures of central tendency are significantly different from 3 ($H_a \neq 3$). The t-test to establish the significance of the measures of central tendency have revealed results that are identical to the non-parametric equivalent test which is the one-sampled Wilcoxon signed rank test illustrated in Table 4.9

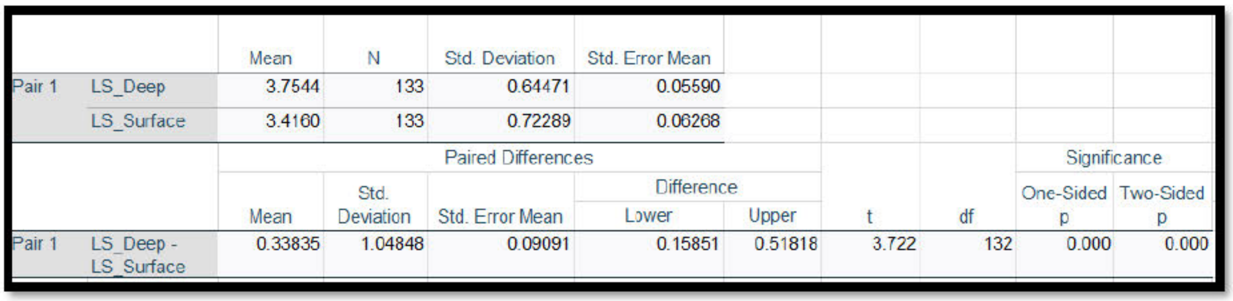
Table 4.9 One Sampled Wilcoxon Signed Rank Test for Learning Styles





As can be observed in Table 4.9 the observed medians were significantly greater than the hypothesised median of 3 for 5 of the 6 questionnaire items (83%). This outcome is consistent with the levels of significance reported via the t-test as well. The implication from this analysis is that there is a significant ($p<0.05$) tendency by the respondents to opt for responses that indicate high levels of deep learning. Another significant observation is that 2 of the 3 questionnaire items that were positively worded to indicate surface learning were also significantly ($p<0.05$) greater than the hypothesised median of 3. While the responses pertaining to deep learning are indicative of a genuine desire to master the challenge pertaining to attaining learning of computer programming, the high scores reported for surface learning are indicative of an intention from the study's respondents to also ensure that they engage in techniques of learning that empower them with a maximum opportunity to pass computer programming assessment activity. An interesting outcome from this observation is to establish which of the learning styles have a dominant presence. This knowledge will provide a greater understanding of the dominant learning behaviour adopted by respondents when it comes to computer programming. According to Boone and Boone (2012) and Joshi, Kale, Chandel, and Pal (2015), when a Likert scale questionnaire contains multiple questionnaire items that measure a single construct, then these questionnaire items can be combined by making use of measures of central tendency such as a mean or median. In the context of the current data, the means and medians of the datasets for deep (3 Likert scale items) and surface (3 Likert scale items) were compared to establish whether there was a significant difference. The comparison of means was conducted via the paired samples t-test as well as the related samples Wilcoxon Signed Rank test. In both instances, the null hypothesis states that there is no significant difference between the means and medians of the deep and surface learning data sets. However, the t-test and the Wilcoxon Signed Rank test indicate that is a significant (P<0.0.5) difference. The outcomes for both these tests are shown in Table 4.10 and Figure 4.6.

Table 4.10 Paired Samples t-test for Deep and Surface Learning



		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	LS_Deep	3.7544	133	0.64471	0.05590
	LS_Surface	3.4160	133	0.72289	0.06268

		Paired Differences							Significance	
					Difference					
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper	t	df	One-Sided p	Two-Sided p
Pair 1	LS_Deep - LS_Surface	0.33835	1.04848	0.09091	0.15851	0.51818	3.722	132	0.000	0.000

In Table 4.10 the mean value reported for deep learning is significantly greater than the mean value reported for surface learning. This outcome is confirmed in the non-parametric equivalent test that was conducted using the related samples Wilcoxon signed rank test. Figure 4.7 illustrates the difference in the mean values reported for deep and surface learning.

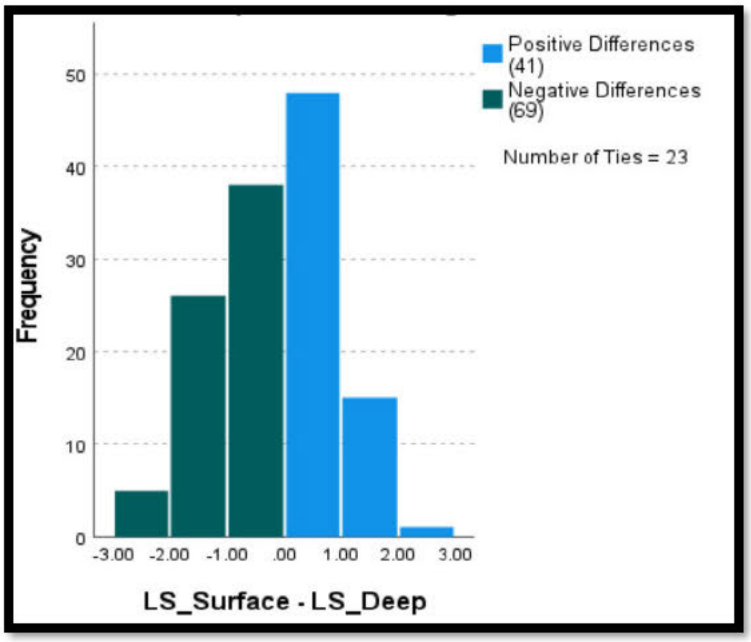


Figure 4.7: Wilcoxon signed rank test for Deep and Surface

From Figure 4.7, it can be seen that 69 of the 133 samples shown a greater median value for deep learning as opposed to 41 responses that favoured surface learning. The outcome of this exercise enables one to confidently make the conclusion that the respondents displayed a significantly greater tendency to adopt a deep learning approach when it comes to academic performance in computer programming. However, respondents also showed a significant preference to adopt a surface learning behaviour when it comes to ensuring academic performance in computer programming.

4.6.4 Frequency Distributions for the Construct Self-Efficacy

The construct of Self-Efficacy (SE) has been represented by 12 questionnaire items where 9 of the items were positively worded in favour of high levels of self-efficacy and 3 questionnaire items were positively worded in favour of low levels of self-efficacy. Essentially the SE construct bears testimony to the cognitive state of respondents as they tackle computer programming tasks. This construct consisted of questionnaire items that were directed as specific aspects of computer programming. These aspects consisted of the ability to write procedural and object-oriented code (4 questionnaire items), the ability to debug and recover from errors and the ability to trace through the logic of computer programming code (2 questionnaire items), the ability to compile a logical computer programming solution to a given problem in a specified time range (4 questionnaire items) and the inclination to seek assistance when it comes to the writing of computer programming solutions (2 questionnaire items)

An overall presentation of the responses is provided in Figure 4.8.

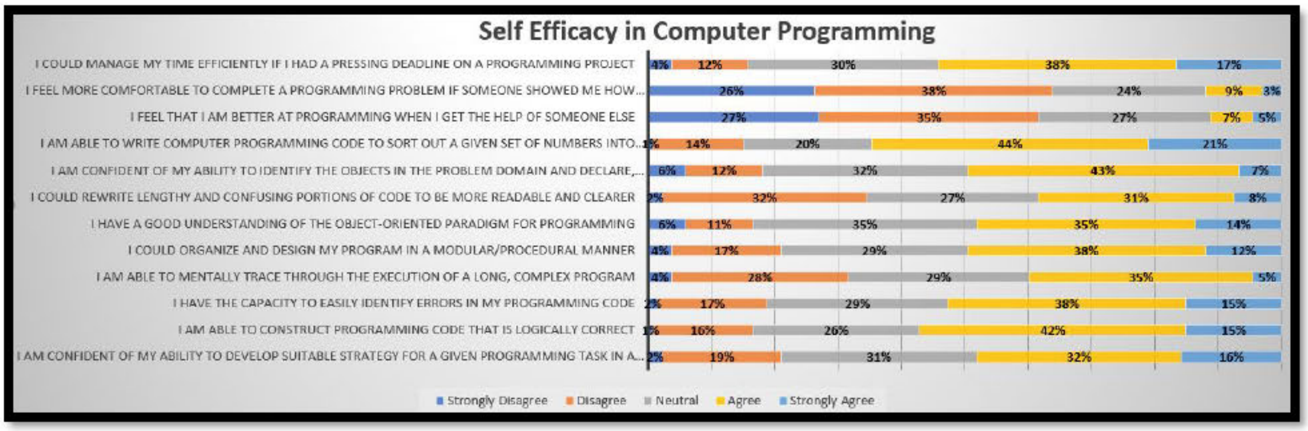
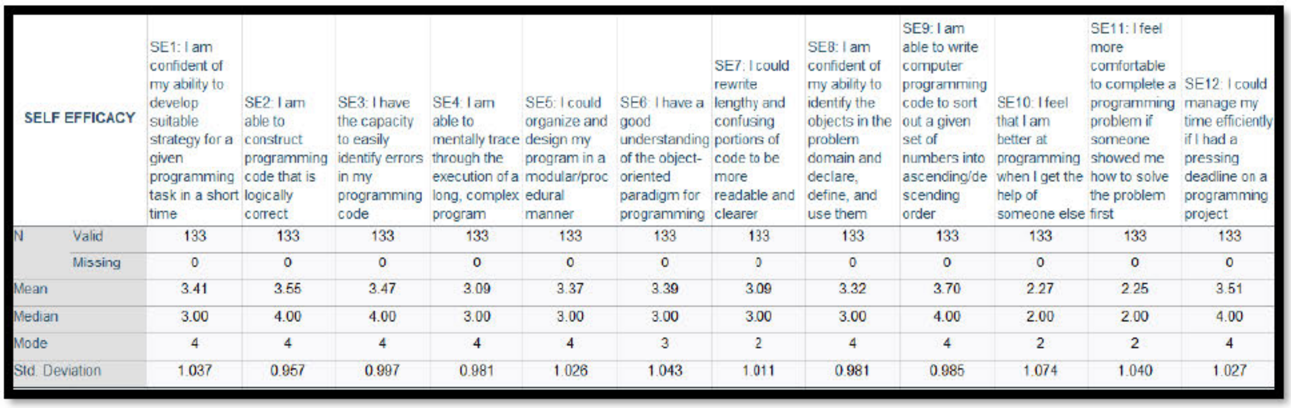


Figure 4.8: Overall view of responses for the construct of Self Efficacy

Ten of the 12 questionnaire items for SE were phrased positively towards the adoption of higher levels of a SE in computer programming. Two of the questionnaire items (11 and 12) were positively worded favouring lower levels of SE. The aggregated outcome of the frequency representation for the SE construct is shown in Table 4.11

Table 4.11 Measures of Central Tendency for Self-Efficacy

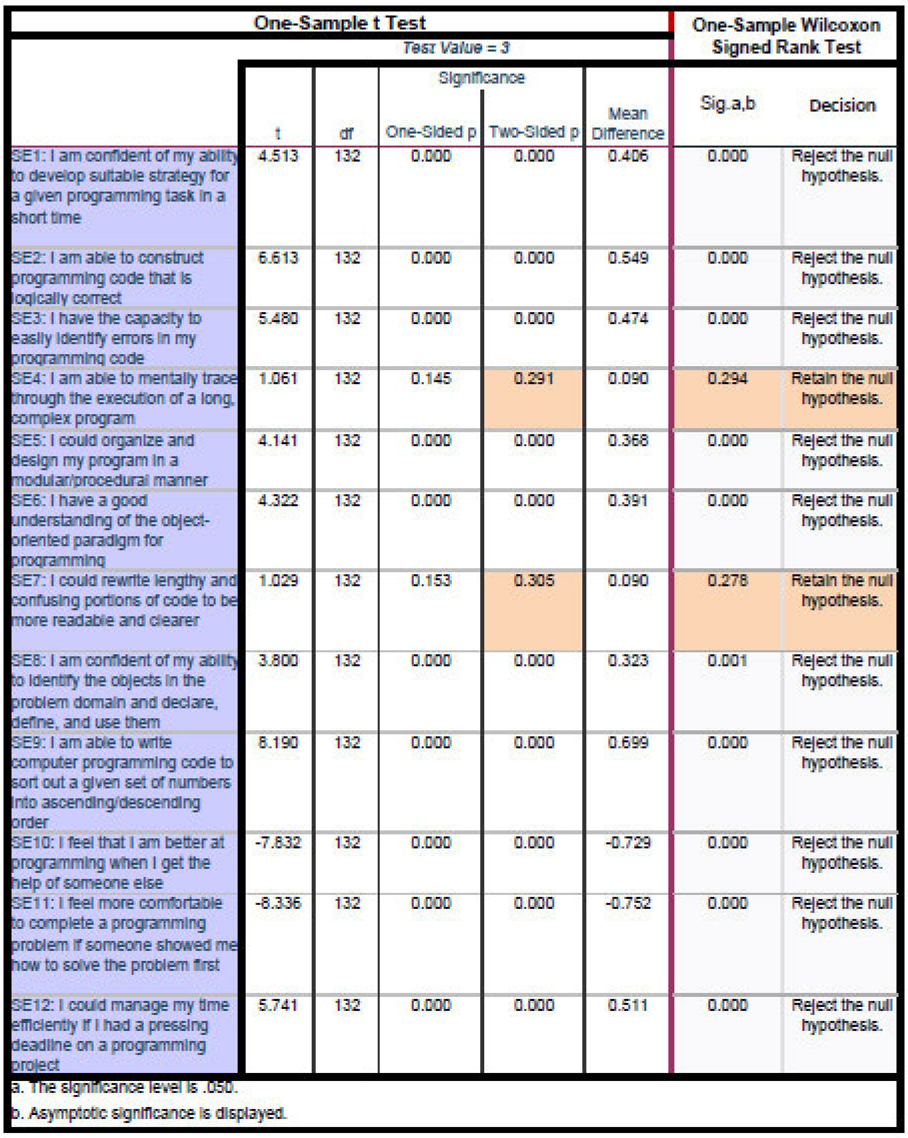


SELF EFFICACY		SE1: I am confident of my ability to develop suitable strategy for a given programming task in a short time	SE2: I am able to construct programming code that is logically correct	SE3: I have the capacity to easily identify errors in my programming code	SE4: I am able to mentally trace through the execution of a long, complex program	SE5: I could organize and design my program in a modular/procedural manner	SE6: I have a good understanding of the object-oriented paradigm for programming	SE7: I could rewrite lengthy and confusing portions of code to be more readable and clearer	SE8: I am confident of my ability to identify the objects in the problem domain and declare, define, and use them	SE9: I am able to write computer programming code to sort out a given set of numbers into ascending/descending order	SE10: I feel that I am better at programming when I get the help of someone else	SE11: I feel more comfortable to complete a programming problem if someone showed me how to solve the problem first	SE12: I could manage my time efficiently if I had a pressing deadline on a programming project
N	Valid	133	133	133	133	133	133	133	133	133	133	133	133
	Missing	0	0	0	0	0	0	0	0	0	0	0	0
Mean		3.41	3.55	3.47	3.09	3.37	3.39	3.09	3.32	3.70	2.27	2.25	3.51
Median		3.00	4.00	4.00	3.00	3.00	3.00	3.00	3.00	4.00	2.00	2.00	4.00
Mode		4	4	4	4	4	3	2	4	4	2	2	4
Std. Deviation		1.037	0.957	0.997	0.981	1.026	1.043	1.011	0.981	0.985	1.074	1.040	1.027

As can be observed in Table 4.11 the mean response is in excess of 3 (*M*>3) and the median is greater than or equal to 3 in 10 of the 12 cases (*Mdn*>=3). In 2 instances, the mean and median response is less than 3. It should be noted that in both of the cases where the mean and median were less than 3, the questionnaire items were phrased positively towards lower levels of SE. The one sample t-test was used to ascertain whether the measures of central tendency were significant or occurred by chance. The results of the one sample t-test as well as the Wilcoxon Signed Rank test are shown in Table 4.12

From Table 4.12 it can be observed that there is a significant difference (p<0.05) between the mean and median values for 10 of the 12 (83%) questionnaire items used to measure SE. Questionnaire items 4 and 7 did not yield results that are significant (p>0.05) and these questionnaire items will be monitored during the correlation phase of the data analysis. Also, the questionnaire items that were positively worded in favour of high levels of SE showed a significant positive difference from the hypothesised neutral values for the mean and median. This outcome is indicative of a high level of SE being displayed by the respondents of the study towards the handling of computer programming tasks. The preceding outcome is further corroborated by the negative differences recorded for Questionnaire items 10 and 11. These questionnaire items were positively worded to indicate low levels of SE. The low means and medians recorded are an indication that the respondents disagreed with the statements attesting to low levels of SE when it comes to academic performance in computer programming.

Table 4.12 Significance Testing for Self-Efficacy



	One-Sample t Test					One-Sample Wilcoxon Signed Rank Test	
	Test Value = 3						
			Significance				
	t	df	One-Sided p	Two-Sided p	Mean Difference	Sig.a,b	Decision
SE1: I am confident of my ability to develop suitable strategy for a given programming task in a short time	4.513	132	0.000	0.000	0.406	0.000	Reject the null hypothesis.
SE2: I am able to construct programming code that is logically correct	6.613	132	0.000	0.000	0.549	0.000	Reject the null hypothesis.
SE3: I have the capacity to easily identify errors in my programming code	5.460	132	0.000	0.000	0.474	0.000	Reject the null hypothesis.
SE4: I am able to mentally trace through the execution of a long, complex program	1.061	132	0.145	0.291	0.090	0.294	Retain the null hypothesis.
SE5: I could organize and design my program in a modular/procedural manner	4.141	132	0.000	0.000	0.368	0.000	Reject the null hypothesis.
SE6: I have a good understanding of the object-oriented paradigm for programming	4.322	132	0.000	0.000	0.391	0.000	Reject the null hypothesis.
SE7: I could rewrite lengthy and confusing portions of code to be more readable and clearer	1.029	132	0.153	0.305	0.090	0.278	Retain the null hypothesis.
SE8: I am confident of my ability to identify the objects in the problem domain and declare, define, and use them	3.800	132	0.000	0.000	0.323	0.001	Reject the null hypothesis.
SE9: I am able to write computer programming code to sort out a given set of numbers into ascending/descending order	8.190	132	0.000	0.000	0.699	0.000	Reject the null hypothesis.
SE10: I feel that I am better at programming when I get the help of someone else	-7.632	132	0.000	0.000	-0.729	0.000	Reject the null hypothesis.
SE11: I feel more comfortable to complete a programming problem if someone showed me how to solve the problem first	-8.336	132	0.000	0.000	-0.752	0.000	Reject the null hypothesis.
SE12: I could manage my time efficiently if I had a pressing deadline on a programming project	5.741	132	0.000	0.000	0.511	0.000	Reject the null hypothesis.

a. The significance level is .050.

b. Asymptotic significance is displayed.

4.7 Problem-Solving Ability and Performance in Computer Programming

The construct of problem ability was operationalised/measured by adapting the computer programming aptitude test used by Tukiainen and Mönkkönen (2002) in predicting computer programming competence. The test to measure computer programming competence was presented to the study's respondents via a series of problem-solving tasks that tested their cognitive processing ability when faced with computer programming related questions. These tasks were adapted to align with the computer programming content that was delivered to the respondents of the current study during their tenure as students of the IS&T curriculum at UKZN. The classification number of questionnaire items used for the problem-solving construct is presented in Table 4.13

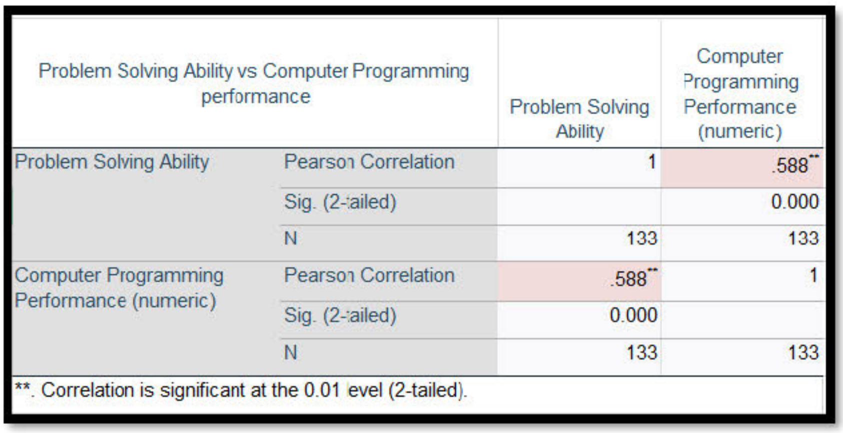
Table 4.13 Classification of Questionnaire items for problem solving

Computer Programming Concept	Number of Questionnaire Items
Conditional logic (logical operators)	2
Predictive logic	3
Comparative logic	1
Iterative logic	2
Assignment logic	1
Data Structure logic	1

Respondents of the study were presented with the set of 10 computer programming related tasks listed in Table 4.12 and were required to provide a response that was structured as a multiple-choice type of question. Students were required to answer these programming related tasks during their computer programming practical sessions at the University. Student were given the latitude of completing these questions without any time restrictions and this resulted in the submission of many late responses. Each of the study's respondents were scored on their performance by allocating a point value of 1 for a correct answer and 0 for an incorrect answer. In this way, each respondent scored a mark out of 10, thereby providing a quantified indicator of the problem-solving ability of the student.

The study's respondents were also required to provide an approximate measure of their academic performance in computer programming assessment (refer to Section A, Part 3 of the Questionnaire in Appendix A). These values were recorded using a scale of 1 to 8. A bivariate correlation analysis was conducted between the respondents' academic performance and their problem-solving ability. The results are presented in Table 4.14.

Table 4.14 Academic performance in computer programming vs Problem Solving ability



Problem Solving Ability vs Computer Programming performance		Problem Solving Ability	Computer Programming Performance (numeric)
Problem Solving Ability	Pearson Correlation	1	.588**
	Sig. (2-tailed)		0.000
	N	133	133
Computer Programming Performance (numeric)	Pearson Correlation	.588**	1
	Sig. (2-tailed)	0.000	
	N	133	133

**. Correlation is significant at the 0.01 level (2-tailed).

According to Saunders et al. (2009) the Pearson Product Moment Co-efficient (PPMC) should be used to assess the validity of a correlation between 2 variables if both the variables are numeric. As can be seen in Table 4.13, the PPMC is statistically significant (r=0.59, n=133, p<0.01, two-tailed). The interpretation from this result is that there is a significantly positive relationship between the respondents' academic performance in computer programming assessment and their problem-solving ability in the context of computer programming tasks. This result provides a measure of validity to the construct of academic performance which is an estimated value provided by the study's respondents.

4.8 Correlation Analysis

The data analysis that has been presented thus far has been used to create a context for the study's data and will contribute to the discussion of results. However, the main focus of the study is to establish the validity of correlations between the study's main variables as well as the study's conceptual framework. The trajectory of the correlation analysis is guided by Musil, Jones, and Warner (1998) in the sequence listed:

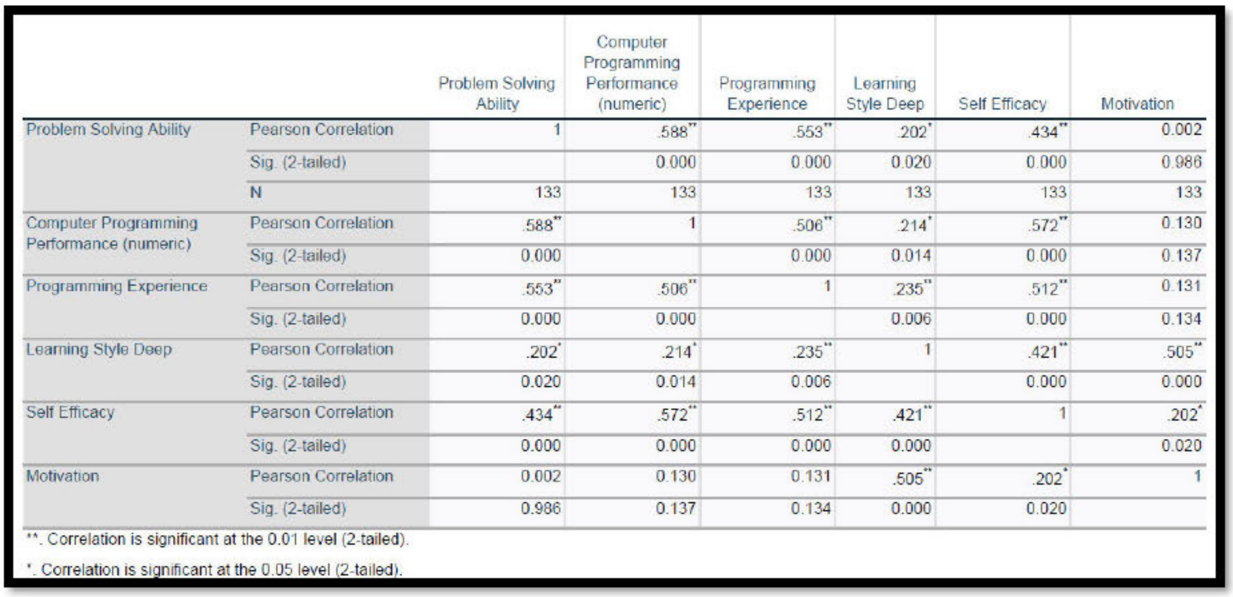
- Correlation analysis – ascertain if there is a statistically significant association between variables of the study
- Regression analysis – this is similar to correlation analysis; however, one or more variables is/are identified as the independent (predictor) variable(s) and that has a mathematically linear relationship with a dependent (outcome) variable that enables one to identify direct causal paths between the independent variables and the dependent variable

- Path Analysis – this is regarded as an extension to multiple regression analysis that allows for the identification of possible indirect causal paths to the dependent variable.

4.8.1 Bivariate Correlation Analysis

The data represented the study's main constructs is represented by ordinal scales and according to Saunders et al. (2009) the PPMC may be used to determine the relationship between these constructs. The Pearson correlation analysis was chosen to establish the significance of relationships between the study's main constructs. It should be noted that the initial matrix of bivariate correlations showed the construct of Learning Styles (LS) to have very poor significance with regards to correlations with the study's other variables. It was decided to use the questionnaire items that were positively worded to ascertain levels of deep learning towards the attainment of good academic performance in computer programming for the generation of the bivariate correlation matrix shown in Table 4.15

Table 4.15 Bivariate Correlation of the study's main constructs



		Problem Solving Ability	Computer Programming Performance (numeric)	Programming Experience	Learning Style Deep	Self Efficacy	Motivation
Problem Solving Ability	Pearson Correlation	1	.588**	.553**	.202*	.434**	0.002
	Sig. (2-tailed)		0.000	0.000	0.020	0.000	0.986
	N	133	133	133	133	133	133
Computer Programming Performance (numeric)	Pearson Correlation	.588**	1	.506**	.214*	.572**	0.130
	Sig. (2-tailed)	0.000		0.000	0.014	0.000	0.137
Programming Experience	Pearson Correlation	.553**	.506**	1	.235**	.512**	0.131
	Sig. (2-tailed)	0.000	0.000		0.006	0.000	0.134
Learning Style Deep	Pearson Correlation	.202*	.214*	.235**	1	.421**	.505**
	Sig. (2-tailed)	0.020	0.014	0.006		0.000	0.000
Self Efficacy	Pearson Correlation	.434**	.572**	.512**	.421**	1	.202*
	Sig. (2-tailed)	0.000	0.000	0.000	0.000		0.020
Motivation	Pearson Correlation	0.002	0.130	0.131	.505**	.202*	1
	Sig. (2-tailed)	0.986	0.137	0.134	0.000	0.020	

**. Correlation is significant at the 0.01 level (2-tailed).

*. Correlation is significant at the 0.05 level (2-tailed).

From Table 4.15 it can be observed that:

- There exists a *moderate*, but positively significant correlation between Problem Solving Ability and Computer Programming Performance **(r=0.59, n=133, p<0.01, two-tailed)**
- There exists a *moderate*, but positively significant correlation between Problem Solving Ability and Computer Programming Experience **(r=0.55 n=133, p<0.01, two-tailed)**

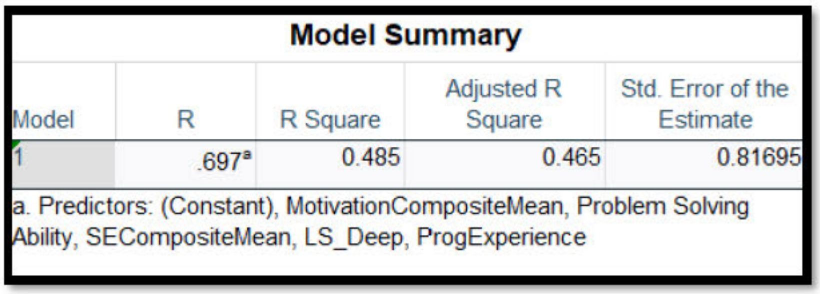
)

- There exists a *weak*, but positively significant correlation between Problem Solving Ability and Learning Styles (deep learning) **(r=0.20 n=133, p<0.01, two-tailed)**
- There exists a *moderate*, but positively significant correlation between Problem Solving Ability and Self Efficacy **(r=0.43, n=133, p<0.01, two-tailed)**
- The results in Table 4.14 indicate that the bivariate correlation between Problem Solving Ability and Motivation is ***not significant* co**

4.8.2 Multiple Regression Analysis (MRA)

As explained in Swanson and Holton (2005) the next step after bivariate correlations is to examine the combined effect of multiple independent variables with the dependent variable. The objective of multiple regression is to provide the researcher with empirical evidence to make decisions regarding the predictive capacity of the study's conceptual model or to enable an explanation of the relationship between the independent and dependent variables in the study. The decision to conduct MRA is informed by Tranmer and Elliot (2008) who suggest that the relationship between the explanatory variables and the dependent variables should be a reasonable one and preferably greater than 0.15. The correlation coefficients recorded in Table 4.14 are greater than .15 although the constructs of learning styles and motivation are quite close to the threshold value of 0.15. These constructs will be monitored in the MRA analysis and the Path analysis that will follow.
In the context of the data for the current study, the multiple regression model is guided by Sekaran and Bougie (2016). The 1st output from this analysis is the Model Summary output (Table 4.16) and the analysis of variance (ANOVA) output (Table 4.17.

Table 4.16 Model Summary for MRA

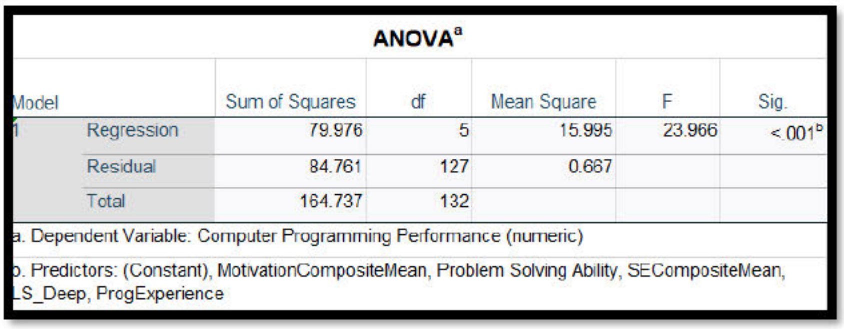


Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.697[a]	0.485	0.465	0.81695

a. Predictors: (Constant), MotivationCompositeMean, Problem Solving Ability, SECompositeMean, LS_Deep, ProgExperience

Table 4.17 ANOVA showing the Significance of the Model



ANOVA[a]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	79.976	5	15.995	23.966	<.001[b]
	Residual	84.761	127	0.667		
	Total	164.737	132			

a. Dependent Variable: Computer Programming Performance (numeric)

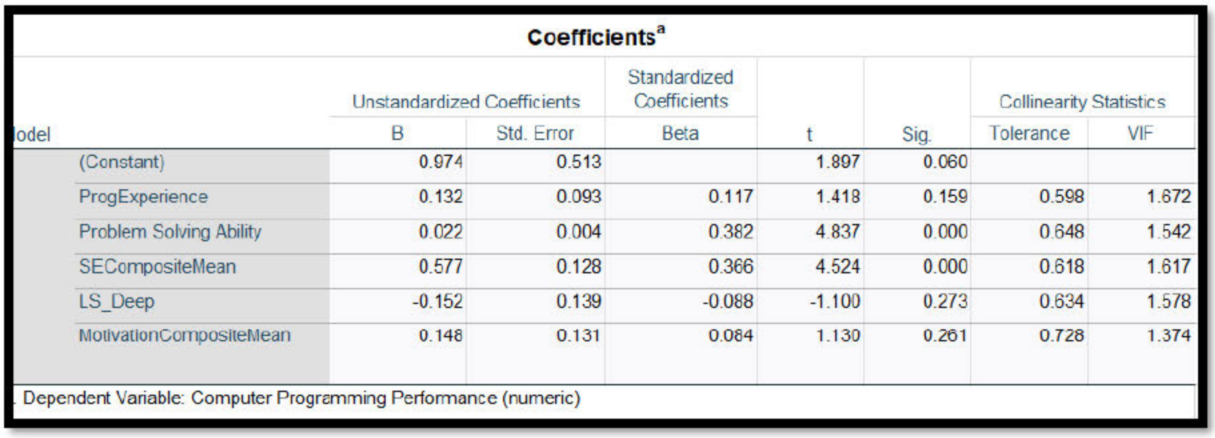
b. Predictors: (Constant), MotivationCompositeMean, Problem Solving Ability, SECompositeMean, LS_Deep, ProgExperience

By analyzing tables 4.16 and 4.17 it can be established that the combined independent variables significantly predict computer programming performance. The statistics that support this conclusion is listed below:

- The F-statistic in Table 4.16 – F(5)=23.96 ($p<0.01$) – is an indicator of the significance of the study's multiple regression model
- The F-statistic provides a validation indicator for the conclusion that the composite set of independent variables explains 46.5% of the variance in the dependent variable ($R=0.7$; $R^2=0.48.5$; adjusted $R^2=0.46.5$; $p<0.01$).

The final multiple regression output is to examine the coefficient values listed in Table 4.18 where it can be seen from the Beta values that problem solving ability and self-efficacy are the 2 main predictors of computer programming performance ($p<0.01$).

Table 4.18 Coefficients of the Multiple Regression Model



Coefficients[a]

		Unstandardized Coefficients		Standardized Coefficients			Collinearity Statistics	
odel		B	Std. Error	Beta	t	Sig.	Tolerance	VIF
	(Constant)	0.974	0.513		1.897	0.060		
	ProgExperience	0.132	0.093	0.117	1.418	0.159	0.598	1.672
	Problem Solving Ability	0.022	0.004	0.382	4.837	0.000	0.648	1.542
	SECompositeMean	0.577	0.128	0.366	4.524	0.000	0.618	1.617
	LS_Deep	-0.152	0.139	-0.088	-1.100	0.273	0.634	1.578
	MotivationCompositeMean	0.148	0.131	0.084	1.130	0.261	0.728	1.374

. Dependent Variable: Computer Programming Performance (numeric)

It should also be noted that according to the data presented in Table 4.18 programming experience, learning styles and motivation are not significant predictors of computer programming performance.

4.8.3 Path Analysis

The preceding correlation analyses looked at direct correlations between the study's set of independent variables and the dependent variable. The objective of path analysis is to examine the correlations based on the study's conceptual model (Musil et al., 1998). The 1st path analysis diagram generated by the study's data is referred to as the "just identified" (Streiner, 2005, p. 6) version where there is a perfect reproduction of the multiple correlation matrix and designed according to the study's conceptual model is shown in Figure 4.9.

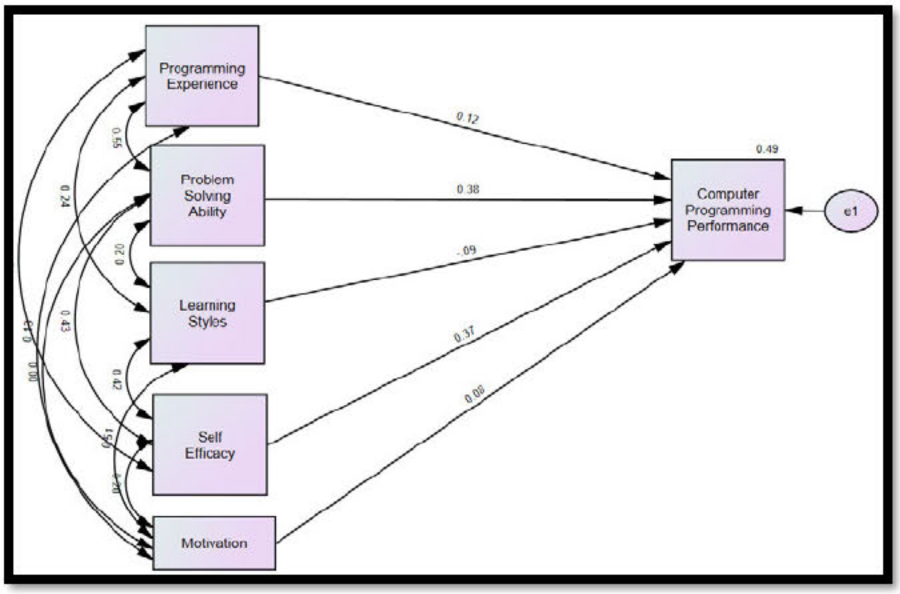


Figure 4.9: Just Identified Path Analysis model

According to Malkanthie (2015) there are various indicators that provide information on the predictive capacity of the model generated by path analysis. The "just identified" model does not consider the possibility of the influence of intervening variables in establishing a set of correlations with the dependent variable. According to Streiner (2005) there are numerous paths between the exogenous (dependent) variables and the endogenous variables. On the basis of path manipulation, one such path model that has been discovered that satisfies the requirements of a "good fit" model is presented in Figure 4.10.

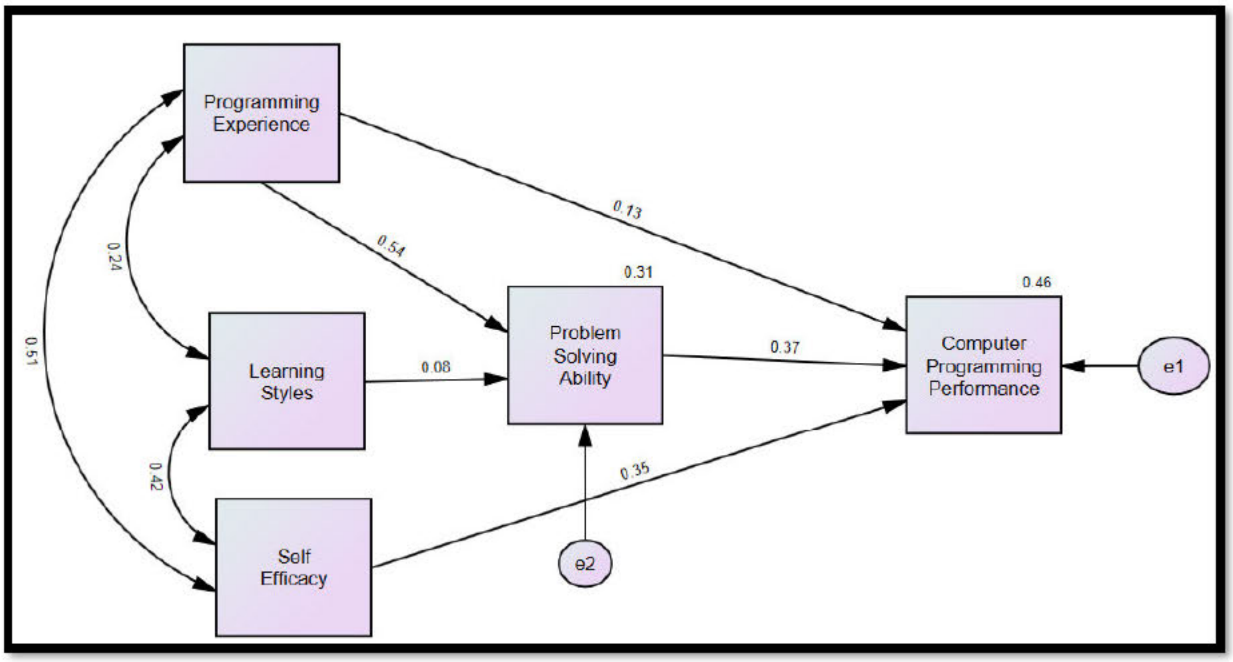


Figure 4.10: Alternative Path Analysis model

In Figure 4.10, an alternative path analysis model has been generated where the model fit indices (GFI=0.95; CFI=0.916; TLI=0.915; RMSEA= 0.113) are indicative of a good model fit. It should be noted that in this model, the construct of Motivation has been removed to ensure that the model is able to account for at least 46% of the variance in the endogenous variable. Based on the study's data, the construct of Motivation did not contribute towards an ideal predictive model and the constructs of Problem-Solving ability and Self Efficacy were the greatest contributors in enabling the model to account for variances in the dependent variable.

4.9 Open Ended Question Analysis

According to Rouder, Saucier, Kinder, and Jans (2021), open-ended survey responses complement the closed-ended section of a survey by providing the respondents of a study with an opportunity to express an opinion in an "open-text" format. These types of questions are used as a follow-up to survey-based questions where the survey section may not obtain the deep insight required for the understanding of a phenomenon. For the current study, a single open ended question was included in the questionnaire where respondents are asked to provide an opinion on their academic performance in computer programming. It should be noted that according to Rouder et al. (2021) the open-ended question could exist on its own and be a full replacement for the close-ended questions especially in situations where the close-ended questions does not fully capture the essence of the topic under inquiry. In the case of academic performance in computer programming, the presence of the open-ended question is crucial because it provides the respondents of the study with an opportunity to respond in a manner that has not been catered for in the close-ended section of the survey. A total of 47 of the 133 respondents took the opportunity to provide responses in an open-ended manner. This represented a response rate of 35%.

The data generated in the open-ended section of the current study consisted of textual data that conveyed diverse opinions on the experience of computer programming. According to Saldaña (2013) a viable technique to regain a measure of control when confronted with a diverse dataset is to use a technique referred to as coding. These codes are used to classify data that are similar in meaning into clusters that are regarded as a conflation/reduction of the original dataset. Stuckey (2015) does advise however, that generating a set of initial codes is not easy and can be guided by the study's main research objective. The main objective for the current study was to obtain empirical evidence attesting to the significance/role played by various factors in predicting computer programming performance. From an overview perspective the study did commence with an a priori model of factors that could influence academic performance in computer programming. According to Stuckey (2015) the constructs from such a model need to be combined with knowledge of the overall objective(s) of the study so that the codes that are identified could be aligned to the structural components of a study. Saldaña (2013) does however assert that the 1st phase of coding may be regarded as quite primitive and could be refined into a 2nd set of codes that have a greater alignment with the study's parameters.

In the current study, the open-ended responses were collated into a spreadsheet application and then imported into the computer assisted qualitative data analysis software application named NVivo (version 12). A word cloud for the initial dataset was generated in Nvivo and presented in Figure 4.11.

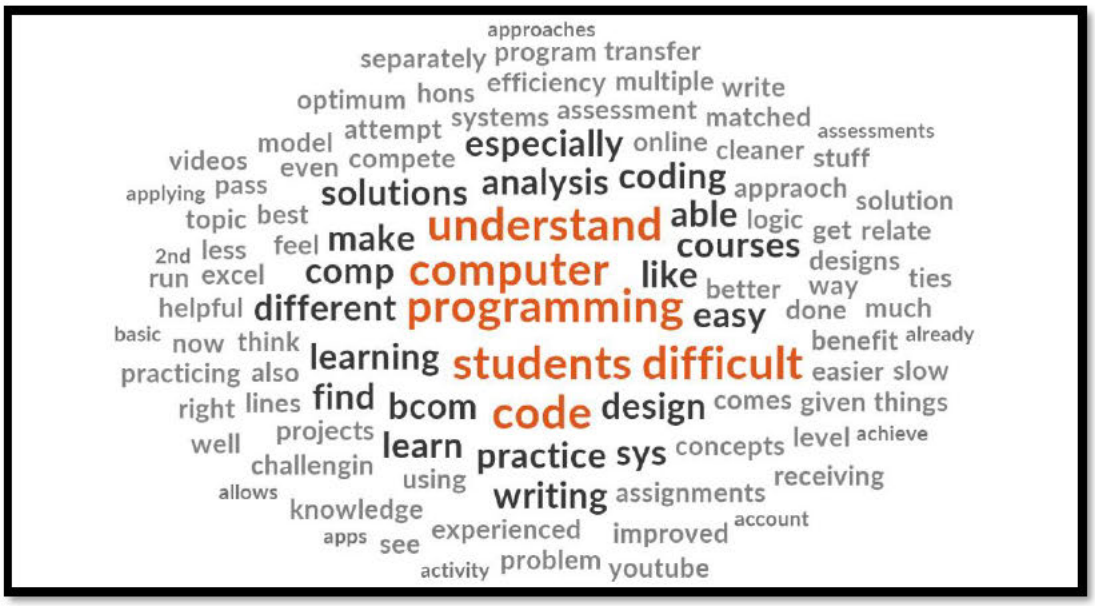


Figure 4.11: Word Cloud for Open Ended Responses

According to Saldaña (2013) the most commonly used words (as shown in Figure 4.11) may be used to generate a rich textural description of the essence of the qualitative data. From Figure 4.11 one of the main themes that may be identified is that "…*students find it difficult to write/understand/learn computer programming code and this could be a major challenge for BCom students. Also, they require extra practice in coding"*. Equipped with this knowledge an initial set of codes was developed and a frequency of items that could be classified into each of the codes is presented in Figure 4.12.

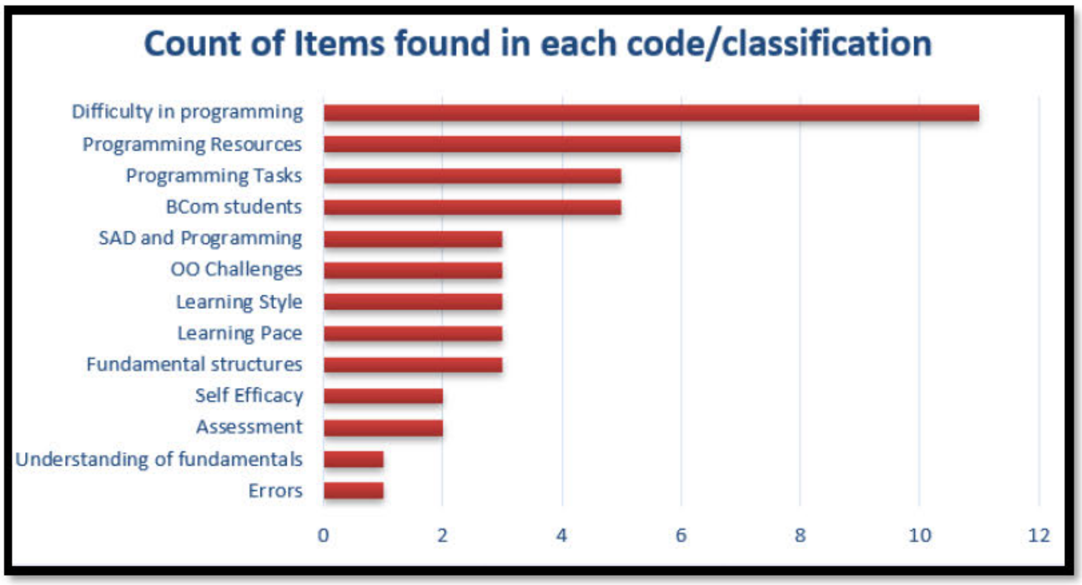


Figure 4.12: Frequency of items in each code

From Figure 4.12 it can be established that the most prominent response was aligned to the difficulties associated with achieving good academic performance in computer programming. Saldaña (2013) advises that the major category of responses provide the researcher with guidance on how to refine the initial set of codes into themes. Based on the guidance provided in Figure 4.12, the main themes that have been identified in the corpus of qualitative data is listed as:

- Difficulty in computer programming
- Access to computer programming resources (conflated with programming tasks)
- The challenges faced by BCom students

While the 1st two themes have been identified by virtue of the number (frequency) of references (confirmed in Figure 4.12), the 3rd theme that refers to the challenges faced by BCom students needs a bit of elaboration. In the open-ended responses, there were 5 instances where students made the claim of being BCom students (as shown in Figure 4.12). Upon further examination of these responses, it was found that in all 5 instances, the students referred to the challenges that BCom students are faced with when it comes to the learning of computer programming.

4.9.1 The difficulty in computer programming

There was a high proportion of responses attached to this theme. A sample of verbatim responses are presented and discussed:

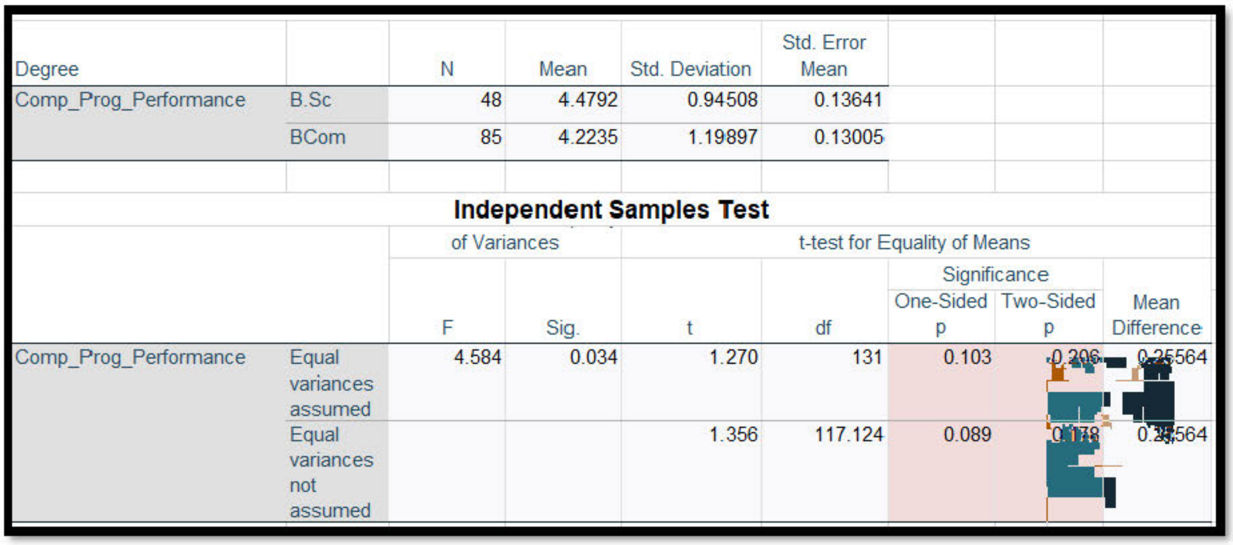
Writing computer programming code is not easy especially if you are not experienced in coding (Theme1; Ref: 6)

This comment alludes to the role that is played by previous experience. The influence of previous experience on computer programming skill/performance has been explored quantitatively in the 1st part of this chapter. This evidence corroborates the significance of previous experience. A further response aligned to this theme is:

Mastering computer programming logic is not easy to achieve (Theme1 ;Ref:8)
It is difficult for BCom students to compete with IT students (Theme1; Ref:11)

These verbatim responses have a direct linkage to the data analysis from the quantitative section of the study where it has been established in Figure 4.6 that respondents who are registered for a BCom degree have lesser experience in computer programming thereby compromising their academic performance in comparison to students who are registered for an IT degree (B.Sc). This qualitative data becomes a catalyst for a further exploration of the quantitative data to establish whether there is a significant difference in computer programming performance and problem solving between BCom students and B.Sc students. The results of the comparative analysis is presented in Table 4.19 From Table 4.18 it can be seen that the mean computer programming performance value for B.Sc students is higher than BCom students. However, independent samples t-test shows that this difference in the mean value is *not* significant. A similar analysis of means was conducted in respect of problem-solving ability and the results are presented in Table 4.20

Table 4.19 Programming Performance for BSc vs BCom

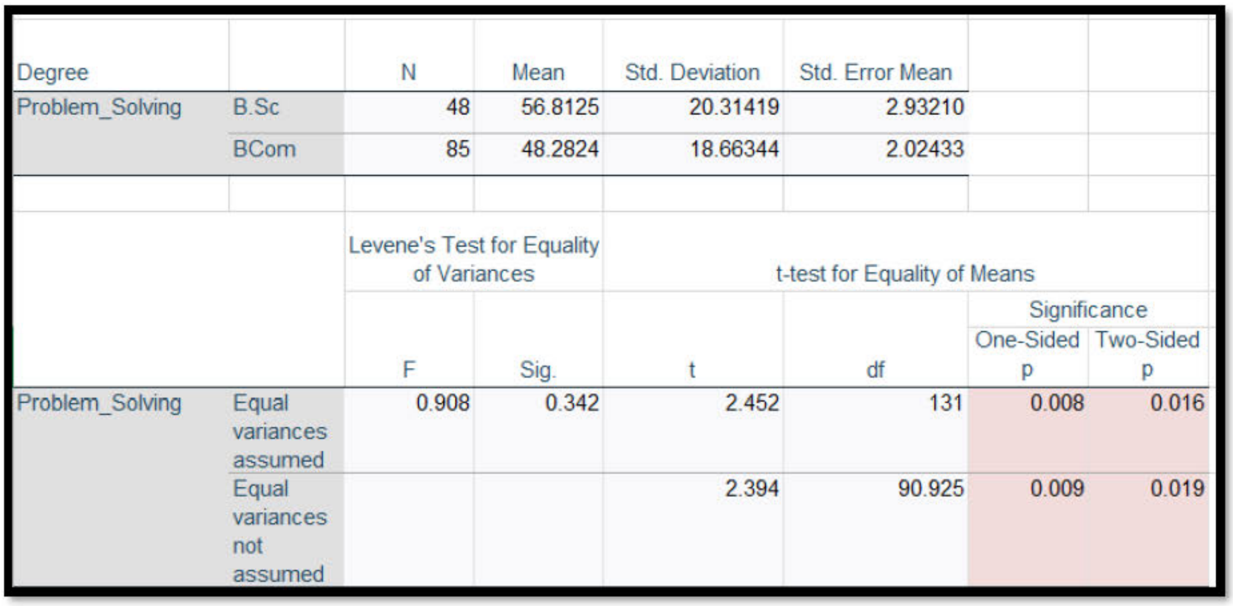


Degree		N	Mean	Std. Deviation	Std. Error Mean
Comp_Prog_Performance	B.Sc	48	4.4792	0.94508	0.13641
	BCom	85	4.2235	1.19897	0.13005

Independent Samples Test

		of Variances		t-test for Equality of Means				
						Significance		
		F	Sig.	t	df	One-Sided p	Two-Sided p	Mean Difference
Comp_Prog_Performance	Equal variances assumed	4.584	0.034	1.270	131	0.103	0.206	0.25564
	Equal variances not assumed			1.356	117.124	0.089	0.178	0.25564

Table 4.20: Problem-solving for BSc vs BCom



Degree		N	Mean	Std. Deviation	Std. Error Mean
Problem_Solving	B.Sc	48	56.8125	20.31419	2.93210
	BCom	85	48.2824	18.66344	2.02433

		Levene's Test for Equality of Variances		t-test for Equality of Means			
						Significance	
		F	Sig.	t	df	One-Sided p	Two-Sided p
Problem_Solving	Equal variances assumed	0.908	0.342	2.452	131	0.008	0.016
	Equal variances not assumed			2.394	90.925	0.009	0.019

From Table 4.20 it can be seen that there is a significant difference in the mean scores obtained for problem solving ability between the BSc students and the BCom students ($M_{BSc} > M_{BCom}$, N=133, p<0.05).

4.9.2 Access to Computer Programming Resources

Many respondents complained about an access to a lack of quality resources to provide guidance that will ensure the attainment of academic performance in computer programming, Verbatim responses such as…

more videos will be enough - more exercises would be good (Theme2; Ref:2)

For different sections in programming, I would like extra learning material, like YouTube link videos to get a better understanding (Theme2; Ref:3)

The reference to video as a means of providing computer programming tuition is quite prominent. This value of this type of tuition has been under-estimated and not viewed with much priority. These comments attest to the importance of enhancing pedagogy in this regard. Also, respondents did make a request for greater practical engagement with computer programming tasks so that their level of knowledge could be improved from a "hands-on" perspective.

we would like more programming practicals - that way we can practice more and understand more (Theme2; Ref:4)

4.9.3 The Challenges faced by BCom students

This theme has been analysed and the quantitative empirical evidence suggests that it is a real problem. It should however be noted that BCom students' performance in computer programming assessment is not significantly different from BSc students. This may be attested to by the fact that surface learning traits may ensure that students are able to secure a pass when it comes to computer programming assessment. However, the BCom students lack of deep understanding of fundamental computer programming concepts has been exposed by the problem-solving tasks where there was a significant difference in performance when compared to BSc students. Also, the analysis in Table 4.2 suggests that BCom students have a significantly lesser amount of computer programming experience than BSc students. This significant difference did not however play a role in determining computer programming performance which was elaborated upon in the quantitative data analysis. The verbatim responses by BCom students in this regard do however highlight their plight when it comes to computer programming.

I feel that more should be done for BCom students programming projects/assignments should be given to us (Theme3; Ref:1)

I wish there was better support for BCom students when it comes to programming (Theme3; Ref:4)

It is difficult for BCom students to compete with IT students (Theme3; Ref5)

4.10 Data Analysis Conclusion

In conclusion the study's data has been presented in a descriptive and inferential manner. The descriptive presentation consisted of tables with measures of central tendency (mean and median) provided. This was supplemented by visualisations of the study's data so that a proper context for the correlation analysis could be created. The correlation analysis conducted has positioned the study to engage in a discussion of results that will enable the answering of the research questions. The open-ended question analysis provided a glimpse of how the "depth-drive" qualitative insight could be integrated with quantitative data analysis techniques to add greater value to the discussions that will prevail in the study's final chapter.

CHAPTER 5 – Discussion and Conclusion

5.1 Introduction

This chapter represents a reconciliation between the study's research questions and objectives, the conceptual model, the empirical analysis and the main outcomes. This chapter is also used as an opportunity to contextualise the study's findings with that which is prevalent in the academic literature. There will also be a discussion of the study's main limitations and a discussion of possible avenues of future research in the attainment of good academic performance in computer programming.

The main aim of this study is to determine factors that influence academic performance in computer programming by IS&T students at UKZN. This problem has been operationalised by a set of research questions and a conceptual model to provide guidance on the answering of the research questions. The constructs in the conceptual model has been identified in the study's literature review thereby adding a measure of construct validity to the conceptual model. The conceptual model's predictive capacity was, however, scrutinised via confirmatory factor analysis. The appropriateness of the model was subjected to further scrutiny by examining the possibility of using the Path Analysis technique to generate a model that had a greater "fit" to the study's data. The final analysis conducted in the study was qualitative and it was necessitated by the open-ended question found in the study's data collection instrument. The study has to be classified as a quantitative study driven by positivistic philosophy. However, the study's open-ended question provided an opportunity to enrich the study's value by catering for aspects of respondent behaviour that has not been predicted. While the intention of the researcher was not to engage in a mixed-methods research, the richness of data in the qualitative section of the study's data collection instrument provided ample opportunity for integrating the quantitative analysis with the qualitative analysis to create a mixed-methods look and feel.
The full interpretation of this study's findings is presented in this chapter in relation to the research questions that are guiding this study.

5.2 The Study’s Context and Findings

The study was commissioned to explore the role that five (5) significant constructs play in academic performance in computer programming. All of these constructs have been identified in the literature section of this study as significant role players in determining academic performance in computer programming. However, these constructs do not act in isolation. There has not been any previous study where all of these constructs have been combined into a single conceptual model. The cogency of this model has been tested by applying confirmatory factor analysis with the result showing that a reasonably “good fit” has been found for the conceptual model and the study’s data. To achieve this “good fit” many of the questionnaire items had to be eliminated and the improvement in the model’s internal reliability is confirmed by an improved set of Cronbach alpha scores that are in excess of 0.7 (shown in Chapter 4, Table 4.4). This strategy of variable reduction to improve the predictive capacity of a model is also used in Chowdhury and Turin (2020) as well as Mai, Tian, Lee, and Ma (2019).

5.2.1 Previous Experience and Computer Programming

The study’s main data collection instrument was a survey that was administered to students who are registered for the IS&T programme spanning from the 2nd level of undergraduate study to Masters. The study’s core set of respondents were students who were studying towards an BCom or a BSc degree where Information Systems was one of the major courses that the student was registered for. While the focus of discussion and data analysis has been centred on the constructs that are found in the study’s conceptual model, the researcher leveraged opportunities to gather data and conduct analyses on periphery constructs that will add value to the discussion of the study’s outcomes. One of these periphery analyses was the correlation between the number of years of computer programming experience and the degree that students were registered for. The findings showed a significant correlation and this knowledge will tie in to the main analysis sections because previous experience was identified as an a priori predictor of computer programming performance. The bivariate correlations did show a moderate but significant, positive correlation. This outcome suggests that previous experience does play significant role in academic performance in computer programming. The results from the current study are consistent with those observed in the systematic literature review by Medeiros, Ramalho, and Falcão (2018) where it was found that in a majority of studies, there is evidence of a positive relationship between previous experience and proficiency in computer programming. However, this is not corroborated by all studies. In the current study, the multiple regression correlation analysis and the subsequent path analysis exercises showed that previous experience did not account for much of the variance in computer programming performance. The implication here is that the case of BCom students being compromised by a lack of previous experience is one that can be circumvented by pedagogical measures that can mitigate against this handicap. The knowledge obtained from the current study regarding previous experience is crucial because the implication is that when previous experience is considered as part of a broader understanding of the factors that influence computer programming proficiency, its significance is quite minimal.

5.2.2 Problem Solving Ability and Computer Programming

One of the study's research questions was to establish the significance of problem-solving ability on proficiency in computer programming. The problem-solving concepts referenced in the current study had a strong computer programming alignment and entailed analogical reasoning, conditional and iteration logic and data structure processing logic. The study's findings are quite convincing. Problem solving ability shows a strong, significant positive correlation with computer programming performance. This outcome is confirmed in the report compiled by Medeiros et al. (2018) where 26 papers on this topic were reviewed. While the significance of problem solving has been confirmed, the claim is made that the instruments used to measure problem solving ability are not consistent and there is no standardised instrument that can be relied upon. The instrument used in the current study was based on the highly recognised IBM programming assessment task (PAT) assessment framework. However, it was adapted to align with the computer programming framework currently used at UKZN. This ensured that students were familiar with the coding fragments to enable a seamless response. Barlow-Jones and van der Westhuizen (2017) used a similar instrument with a similar research agenda at the University of Johannesburg and the outcome was that problem-solving ability was a "major" predictor of computer programming performance. The implication from these results suggest that universities need to invest more time at 1st year level where there is focus on logical reasoning and algorithmic thinking so that students can obtain foundation knowledge on computer programming semantic structures to enhance problem solving.

This observation has significant implications for students who have not had prior experience in computer programming because a focus on algorithmic thinking would equip them with the cognitive structures required to obtain a deep understanding of computer programming logic.

5.2.3 Self-Efficacy (SE) and Computer Programming

The role that SE plays on computer programming performance has not received much prominence in the current literature on computer programming. In the past studies such as those by Bergin and Reilly (2006) and Ramalingam, LaBelle, and Wiedenbeck (2004, June) did show that SE was a good predictor of computer programming performance. This outcome is confirmed in the current study where 9 questionnaire items were used to establish the levels of SE of students when it comes to computer programming. The significant positive correlation observed indicates that SE is a good predictor of computer programming performance. However, as Tsai, Wang, and Hsu (2019) points out there is also a strong positive correlation between SE and previous experience, which is confirmed in the current study. The current study extends the network of influence regarding SE by observing that there is a strong positive correlation between SE and a deep learning style. These observations are significant from a pedagogical perspective because educators should make a concerted effort to enhance and enable high levels of SE amongst students in their programming courses. This can be achieved by providing students with extra resources to enable the attainment of good academic performance in computer programming, more practical tasks and extending the notional time of engagement with computer programming activity. This will mitigate any negative consequence that may accrue due to a lack of prior experience in computer programming or a lack of problem-solving ability. Hence, the construct of SE is actually a function of previous experience and problem solving ability (Ramalingam et al., 2004, June) and it was used in the current study's conceptual model as a dependent variable.

5.2.4 Intrinsic and Extrinsic Motivation and Computer Programming

The construct of motivation played a minimal role in predicting computer programming performance. While this construct had a weak but positive correlation with SE and learning styles it did not display a significant relationship with problem solving ability, previous experience or computer programming performance. This observation is consistent with the results in a similar study by Pawlowski (2007) where it was reported that motivation made a contribution to the bivariate analysis framework but did not appear in the regression model because of a lack of significance. This outcome is contrary to the results reported in the study by Bergin and Reilly (2005) where intrinsic and extrinsic motivation was strongly aligned to computer programming performance. An interesting study is one conducted by Kim and Frick (2011) who observed that when students are confined to online learning, high levels of intrinsic motivation was a predictor of success in an online learning environment.

5.2.5 Open Ended Responses pertaining to computer programming

The study's open-ended question provided the researcher with a different dimension to analyse students' opinion on the topic of computer programming. The thematic analysis revealed reasons for poor performance in computer programming. These were a lack of learning resources, a lack of practical exercises and an apprehension by BCom students towards computer programming assessment. The open-ended responses became a catalyst for further analysis where an independent samples t-test comparison established that there was a significant difference in the problem-solving ability of BSc students in comparison to BCom students. However, this difference was not significant when it came to academic performance in computer programming. The data that was available for analysis indicated that BCom students were able to leverage surface learning characteristics and acquire good results for computer programming assessment. This observation ties into the final construct in the study's conceptual model which is learning approaches/learning styles. The intention here was to measure the influence of learning styles on computer programming performance. The study's survey-based data showed that there was a weak but positive correlation between a deep learning style and computer programming performance and problem-solving ability. The implication here is that students who have adopted a deep learning style generally tended to perform better in computer programming assessment. However, the weak correlation implies that students who adopted a surface approach to learning computer programming were also able to obtain a good performance score in computer programming assessment. This was revealed in the path analysis exercise where the learning styles construct had a minimal influence on the model's ability to account for the variance in computer programming performance. This result suggests that the learning styles construct will have a greater impact in measuring the ability to learn computer programming if the assessment used is adjusted to include questions that cater for a deep learning strategy.

5.3 The Study's Limitations and Delimitations

The main limitation from the study is the threat to the external validity because a greater, more expansive sample would have created an opportunity for greater generalisation of the study's results. The delimitation of confining the study to IS&T students was necessitated by the researcher's concerns when it came to data collection because at the commencement of the study, the COVID-19 pandemic had devastating implications for free and open communication with potential respondents for the study. The researcher's outreach was confined to IS&T platforms that were made available online. Another limitation of the study was the potential breach of internal validity because the measurement of student programming performance was done through an estimate provided by the study's respondent. The original plan was to obtain knowledge of student marks in computer programming assessment via the IS&T department. The Protection of Personal Information (POPI) Act did however prevent access to any personal data pertaining to the study's respondents. This potential weakness in the study was however mitigated by the inclusion of problem-solving tasks into the study's questionnaire. The strong positive correlation between the scores obtained in the problem-solving tasks and the respondents' estimation of their performance in computer programming assessment enhanced the reliability of these variables. The inclusion of problem-solving activity did deter students' participation in the study resulting a lower than expected response rate.

A further limitation of the study is that the researcher did not factor-in the possible influence of additional variables that may have been introduced because of the strategy of online learning that was adopted at UKZN. This limitation provides an avenue for further research on the topic of online learning and its influence on computer programming performance.

5.4 Overall Study Conclusion

This study was aimed at addressing the issue of students struggling to obtain proficiency in the domain of computer programming. There have been numerous previous research efforts that have studied this phenomenon and knowledge around this topic has grown substantially. The problem of poor performance in computer does however continue to prevail. The current study was grounded by the previous efforts at finding a solution to this phenomenon. The difference however, is that this study adopted an inclusive approach where the main factors that influence academic performance in computer programming have been integrated into a single conceptual framework. The empirical analysis activity around this framework enabled the researcher to obtain a better understanding of the challenge of learning to program. This is the first study that has shown the relative importance of each factor in contributing towards an improvement in students' performance in computer programming. The study was able to elucidate the pivotal role played by problem solving ability and self-efficacy in predicting performance in computer programming.

The implication is that factors such as intrinsic and extrinsic motivation, previous programming experience and deep and surface learning do contribute, but only in a peripheral manner.

This knowledge provides great pedagogical insight to lecturers and course coordinators because when students are empowered with a computer programming mindset that is generated through a comprehensive knowledge of computer programming fundamentals where logical, conditional and iterative structures are given substantial focus, then a solid grounding is established for data structure processing and object-oriented programming at a later stage. Also, the open-ended section of the data collection instrument provided great insight into students' experience of learning to program. The plight of the BCOM student has been highlighted quite extensively and it is clear that these students do struggle with analogical reasoning and the quest to obtain deep understanding of computer programming fundamentals. This study highlights the need for a pedagogical intervention based on imparting problem-solving skills to these students.

A further outcome is the development of a conceptual model to predict computer programming performance. This model has been subjected to validity testing in the form of confirmatory factor analysis, multiple regression analysis as well as path analysis. The study did produce a "good fitting" predictive model for the study's data. However, the final model that was identified in the study is not an optimal one and opens up an avenue of extension where structural equation modelling techniques could be used to discover an optimal model that provides a maximum explanation of the variance in computer programming performance. A further interesting area of study is to explore the role that motivation and learning style play when students learn programming in an online environment as compared to a face to face setting.

References

Abdunabi, R., Hbaci, I., & Ku, H. (2019). Towards enhancing programming self-efficacy perceptions among undergraduate Information Systems students. *Journal of Information Technology Education, 18*.

Aivaloglou, E., & Hermans, F. (2019). Early programming education and career orientation: the effects of gender, self-efficacy, motivation and stereotypes. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 679-685.

Alase, A. (2017). The interpretative phenomenological analysis (IPA): A guide to a good qualitative research approach. *International Journal of Education and Literacy Studies, 5*(2), 9-19.

Ali, P. J., Ali, S., & Farag, W. E. (2014). An instrument to measure math attitudes of computer science students. *International Journal of Information and Education Technology, 4*(5), 459.

Amabile, T. M., Hill, K. G., Hennessey, B. A., & Tighe, E. M. (1994). The Work Preference Inventory: assessing intrinsic and extrinsic motivational orientations. *Journal of Personality and Social Psychology 66*(5), 950.

Andriotis, N. (2014). Gamification survey results. *Retrieved from* https://www.talentlms.com/blog/gamification-survey-results/

Apuke, O. (2017). Quantitative research methods: A synopsis approach. *Kuwait Chapter of Arabian Journal of Business Management Review, 33*(5471), 1-8.

Askar, P., & Davenport, D. (2009). An investigation of factors related to self-efficacy for Java Programming among engineering students. *Online Submission - Turkish Online Journal of Educational Technology, 8*(1). Retrieved from https://eric.ed.gov/?id=ED503900

Bain, G., & Barnes, I. (2014). Why is programming so hard to learn? In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, 356-356.

Balmes, I. L. (2017). Correlation of mathematical ability and programming ability of the computer science students. *Asia Pacific Journal of Education, Arts and Sciences, 4*(3), 85-88.

Bandura, A. (2006). Guide for constructing self-efficacy scales. *Self-efficacy Beliefs of Adolescents 5*(1), 307-337.

Bandura, A., & Wessels, S. (1994). Self-efficacy. *Vol 4* 71-81.

Barlow-Jones, G., & van der Westhuizen, D. (2017). Problem solving as a predictor of programming performance. In *ICT Education: 46th Annual Conference of the Southern African Computer Lecturers' Association, SACLA 2017, Magaliesburg, South Africa, July 3- 5 2017, Revised Selected Papers* 46, 209-216. Springer International Publishing.

Bennedsen, J., & Caspersen, M. E. (2007). Assessing process and product: a practical lab exam for an introductory programming course. *Innovation in Teaching and Learning in Information and Computer Sciences, 6*(4), 183-202.

Bergin, S., & Reilly, R. (2005). The influence of motivation and comfort-level on learning to program. *Proceedings of the 17th Workshop of the Psychology of Programming Interest Group* 293-304.

Bergin, S., & Reilly, R. (2006). Predicting introductory programming performance: A multi-institutional multivariate study. *Computer Science Education, 16*(4), 303-323.

Biró, P., & Csernoch, M. (2014). Deep and surface metacognitive processes in non-traditional programming tasks. In *2014 5th IEEE Conference on Cognitive Infocommunications (CogInfoCom)*, 49-54. IEEE.

Boone, H. N., & Boone, D. A. (2012). Analyzing likert data. *Journal of Extension 50*(2), 1-5.

Bujang, M. A., Omar, E. D., & Baharum, N. A. (2018). A review on sample size determination for Cronbach's alpha test: a simple guide for researchers. *The Malaysian Journal of Medical Sciences: MJMS 25*(6), 85.

Butler, M., & Morgan, M. (2007). Learning challenges faced by novice programming students studying high level and low feedback concepts. In *Proceedings Ascilite, Singapore,* 1, 99-107.

Chen, Y., Lasecki, W. S., & Dong, T. (2021). Towards Supporting Programming Education at Scale via Live Streaming. *Proceedings of the ACM on Human-Computer Interaction, 4*(CSCW3), 1-19.

Chowdhury, M. Z. I., & Turin, T. C. (2020). Variable selection strategies and its importance in clinical prediction modelling. *Family Medicine and Community Health 8*(1).

Co, M., & Chu, K. M. (2020). Distant surgical teaching during COVID-19-A pilot study on final year medical students. *Surgical Practice 24*(3), 105-109.

Creswell, J. W., & Creswell, J. D. (2017). *Research design: Qualitative, quantitative, and mixed methods approaches*: Sage publications.

de Araujo, A. L. S. O., Andrade, W. L., & Guerrero, D. D. S. (2016). A systematic mapping study on assessing computational thinking abilities. In *2016 IEEE frontiers in education conference (FIE)*, 1-9. IEEE.

DeCoster, J., & Claypool, H. (2004). Data analysis in SPSS.

Du Boulay, B. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research, 2*(1), 57-73.

Durak, H. Y., Yilmaz, F. G. K., & Yilmaz, R. (2019). Computational Thinking, Programming Self-Efficacy, Problem Solving and Experiences in the Programming Process Conducted with Robotic Activities. *Contemporary Educational Technology, 10*(2), 173-197.

Duran, I. L. (2016). The role of mathematics background in the performance of BSCS students in computer programming subject. *International Journal of Multidisciplinary Research and Modern Education (IJMRME), 2*(1), 147-150.

Edwards, S. H., Murali, K. P., & Kazerouni, A. M. (2019). The Relationship Between Voluntary Practice of Short Programming Exercises and Exam Performance. In *Proceedings of the ACM Conference on Global Computing Education*, 113-119. ACM.

Elshiekh, R., & Butgerit, L. (2017). Using gamification to teach students programming concepts. *Open Access Library Journal, 4*(8), 1-7.

Fang, X. (2012). Application of the participatory method to the computer fundamentals course. *Affective Computing and Intelligent Interaction,* 185-189.

Farrell, A. M., & Rudd, J. M. (2009). Factor analysis and discriminant validity: A brief review of some practical issues. In *ANZMAC 2009,* Anzmac.

Fincher, S., Robins, A., Baker, B., Box, I., Cutts, Q., de Raadt, M., ... & Tutty, J. . (2006). Predictors of success in a first programming course. In *Proceedings of the 8th Australasian Computing Education Conference (ACE 2006)*, 52, 189-196. Retrieved from https://research.usq.edu.au/item/9y1qv/predictors-of-success-in-a-first-programming-course

Floyd, K. S., Harrington, S. J., & Santiago, J. (2009). The Effect of Engagement and Perceived Course Value on Deep and Surface Learning Strategies. *Informing Science,, 12* 181-190.

Forte, A., & Guzdial, M. (2005). Motivation and nonmajors in computer science: identifying discrete audiences for introductory courses. *IEEE Transactions on Education, 48*(2), 248-253.

Garner, S., Haden, P., & Robins, A. (2005). My program is correct but it doesn't run: a preliminary investigation of novice programmers' problems. In *Proceedings of the 7th Australasian conference on Computing education-Volume 42*, 173-180.

Gliem, J. A., & Gliem, R. R. (2003). *Calculating, interpreting, and reporting Cronbach's alpha reliability coefficient for Likert-type scales*. Columbus, Ohio. *Retrieved from* https://hdl.handle.net/1805/344

Goldman, K., Gross, P., Heeren, C., Herman, G., Kaczmarczyk, L., Loui, M. C., & Zilles, C. (2008). Identifying important and difficult concepts in introductory computing courses using a delphi process. *ACM SIGCSE Bulletin, 40*(1), 256-260.

Goodwin, L. D. (1999). The role of factor analysis in the estimation of construct validity. *Measurement in Physical Education and Exercise Science 3*(2), 85-100.

Gottfried, A. E. (1985). Academic intrinsic motivation in elementary and junior high school students. *Journal of Educational Psychology 77*(6), 631.

Govender, D. W., & Basak, S. K. (2015). An investigation of factors related to self-efficacy for java programming among computer science education students. *Journal of Governance and Regulation, 4*(4), 612-619.

Govender, I. (2021). Towards understanding information systems students' experience of learning introductory programming: A phenomenographic approach. *Journal of Information Technology Education: Innovations in Practice, 20* 081-092.

Govender, I., Govender, D. W., Havemga, M., Mentz, E., Breed, B., Dignum, F., & Dignum, V. (2014). Increasing self-efficacy in learning to program: exploring the benefits of explicit instruction for problem solving. *TD: The Journal for Transdisciplinary Research in Southern Africa, 10*(1), 187-200.

Hanson, W. E., Creswell, J. W., Clark, V. L. P., Petska, K. S., & Creswell, J. D. (2005). Mixed methods research designs in counseling psychology. *Journal of Counseling Psychology 52*(2), 224.

Heppner, P. P., & Petersen, C. H. (1982). The development and implications of a personal problem-solving inventory. *Journal of Counseling Psychology 29*(1), 66.

Hoskin, T. (2012). Parametric and nonparametric: Demystifying the terms. *Mayo Clinic 5*(1), 1-5.

Hughes, J., & Peiris, D. R. (2006). ASSISTing CS1 students to learn: learning approaches and object-oriented programming. *ACM SIGCSE Bulletin, 38*(3), 275-279.

Hulleman, C. S. (2007). The Role of Utility Value in the Development of Interest and Achievement. *Online Submission*.

Hurley, A. E., Scandura, T. A., Schriesheim, C. A., Brannick, M. T., Seers, A., Vandenberg, R. J., & Williams, L. J. (1997). Exploratory and confirmatory factor analysis: Guidelines, issues, and alternatives. *Journal of Organizational Behavior* 667-683.

Islam, M. R. (2018). Sample size and its role in Central Limit Theorem (CLT). *Computational and Applied Mathematics Journal, 4*(1), 1-7.

Ismail, M. N., Ngah, N. A., & Umar, I. N. (2010). Instructional strategy in the teaching of computer programming: a need assessment analyses. *The Turkish Online Journal of Educational Technology (TOJET), 9*(2), 125-131.

Jenkins, T. (2002). On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences,* 4 (2002), 53-58.

Joshi, A., Kale, S., Chandel, S., & Pal, D. K. (2015). Likert scale: Explored and explained. *British Journal of Applied Science & Technology 7*(4), 396.

Kalelioglu, F., & Gülbahar, Y. (2014). The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective. *Jourmatics in Education, 13*(1), 33-50.

Kallia, M., & Sentance, S. (2019). Learning to use functions: The relationship between misconceptions and self-efficacy. In *Proceedings of the 50th ACM technical symposium on computer science education,* 752-758.

Kanaparan, G., Cullen, R., & Mason, D. (2019). Effect of self-efficacy and emotional engagement on introductory programming students. *Australasian Journal of Information Systems, 23*.

Karsten, R., Mitra, A., & Schmidt, D. (2012). Computer self-efficacy: A meta-analysis. *Journal of Organizational and End User Computing (JOEUC), 24*(4), 54-80.

Khaleel, F. L., Ashaari, N. S., & Wook, T. S. M. T. (2019). An empirical study on gamification for learning programming language website. *Jurnal Teknologi, 81*(2).

Khaleel, F. L., Ashaari, N. S., & Wook, T. S. M. T. (2020). The impact of gamification on students learning engagement. *International Journal of Electrical and Computer Engineering, 10*(5), 4965.

Khaleel, F. L., Ashaari, N. S., Wook, T. S. M. T., & Ismail, A. (2015). User-enjoyable learning environment based on Gamification elements. In *2015 International Conference on Computer, Communications, and Control Technology (I4CT)*, 221-226. IEEE. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7219570

Khraishi, T. (2021). Personal Experiences from Teaching Virtually Online During the COVID-19 Pandemic. In *ASEE 2021 Gulf-Southwest Annual Conference,*

Kim, K.-J., & Frick, T. W. (2011). Changes in student motivation during online learning. *Journal of Educational Computing Research, 44*(1), 1-23.

Kinnunen, P. (2009). *Challenges of teaching and studying programming at a university of technology-Viewpoints of students, teachers and the university.* (PhD Doctoral). Alto University Learning center, Retrieved from https://aaltodoc.aalto.fi/handle/123456789/4710

Kittur, J. (2020). Measuring the programming self-efficacy of Electrical and Electronics Engineering students. *IEEE Transactions on Education, 63*(3), 216-223.

Kolar, H., Carberry, A., R, & Amresh, A. (2013). Measuring computing self-efficacy. In *2013 ASEE Annual Conference & Exposition, Atlanta, Georgia,* 23-889. Retrieved from https://peer.asee.org/22274

Konecki, M., & Petrlic, M. (2014). Main problems of programming novices and the right course of action. In *Central European Conference on Information and Intelligent Systems, Croatia, Faculty of Organization and Informatics Varazdin,* 116-123.

Kori, K., Pedaste, M., Leijen, Ä., & Tõnisson, E. (2016). The role of programming experience in ICT students' learning motivation and academic achievement. *International Journal of Information and Education Technology, 6*(5), 331.

Kori, K., Pedaste, M., Niitsoo, M., Kuusik, R., Altin, H., Tõnisson, E., . . . Siiman, L. (2015). Why do students choose to study Information and Communications Technology? *Procedia-Social and Behavioral Sciences, 191* 2867-2872.

Kori, K., Pedaste, M., Tõnisson, E., Palts, T., Altin, H., Rantsus, R., . . . Rüütmann, T. (2015). First-year dropout in ICT studies. In *2015 IEEE Global Engineering Education Conference (EDUCON)*, 437-445. IEEE.

Korkmaz, Ö., & Altun, H. (2014). Adapting computer programming self-efficacy scale and engineering students' self-efficacy perceptions. *Participatory Educational Research, 1*(1), 20-31. Retrieved from:https://doi.org/10.17275/per.14.02.1.1

Koulouri, T., Lauria, S., & Macredie, R. D. (2014). Teaching introductory programming: A quantitative evaluation of different approaches. *ACM Transactions on Computing Education (TOCE), 14*(4), 1-28. Retrieved from https://doi.org/10.1145/2662412

Kurkovsky, S. (2006). Improving Student Motivation in a Computing Course for Non-Majors. In *FECS, Las Vegas, Nevada USA*, 82-88.

Law, K. M., Lee, V. C., & Yu, Y.-T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers & Education, 55*(1), 218-228. Retrieved from https://doi.org/10.1016/j.compedu.2010.01.007

Lawan, A. A., Abdi, A. S., Abuhassan, A. A., & Khalid, M. S. (2019). What is Difficult in Learning Programming Language Based on Problem-Solving Skills? In *2019 International Conference on Advanced Science and Engineering (ICOASE)*, 18-22. IEEE.

Levy, Y., & Ellis, T. J. (2006). A systems approach to conduct an effective literature review in support of information systems research. *Informing Science, 9*.

Li, L., Xu, L. D., He, Y., He, W., Pribesh, S., Watson, S. M., & Major, D. A. (2021). Facilitating online learning via zoom breakout room technology: a case of pair programming involving students with learning disabilities. *Communications of the Association for Information Systems, 48*(1), 12.

Lindblom-Ylänne, S., Parpala, A., & Postareff, L. (2018). What constitutes the surface approach to learning in the light of new empirical evidence? *Studies in Higher Education* 1-13.

Lishinski, A., Yadav, A., & Enbody, R. (2017). Students' emotional reactions to programming projects in introduction to programming: Measurement approach and influence on learning outcomes. *Proceedings of the 2017 ACM Conference on International Computing Education Research* 30-38.

Lishinski, A., Yadav, A., Enbody, R., & Good, J. (2016). The Influence of Problem Solving Abilities on Students' Performance on Different Assessment Tasks in CS1. In *Proceedings of the 47th ACM technical symposium on computing science education*, 329-334. ACM.

Loksa, D., Ko, A. J., Jernigan, W., Oleson, A., Mendez, C. J., & Burnett, M. M. (2016). Programming, problem solving, and self-awareness: Effects of explicit guidance. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, 1449-1461.

Lonka, K., Olkinuora, E., & Mäkinen, J. (2004). Aspects and prospects of measuring studying and learning in higher education. *Educational Psychology Review, 16*(4), 301-323.

Luxton-Reilly, A., Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., . . . Szabo, C. (2018). Introductory programming: a systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, 55-106. ACM.

Mahatanankoon, P., & Wolf, J. (2021). Cognitive Learning Strategies in an Introductory Computer Programming Course. *Information Systems Education Journal, 19*(3), 11-20.

Mai, F., Tian, S., Lee, C., & Ma, L. (2019). Deep learning models for bankruptcy prediction using textual disclosures. *European Journal of Operational Research 274*(2), 743-758.

Malik, S. I., Shakir, M., Eldow, A., & Ashfaque, M. W. (2019). Promoting Algorithmic Thinking in an Introductory Programming Course. *International Journal of Emerging Technologies in Learning, 14*(1).

Malkanthie, A. (2015). *Structural Equation Modeling with AMOS*. Retrieved from:http://dx.doi.org/10.13140/RG.2.1.1960.4647

Maltby, J. R., & Whittle, J. (2000). Learning programming online: Student perceptions and performance. In *Proceedings of the ASCILITE 2000 Conference*,

Marton, F., & Säljö, R. (1976). On qualitative differences in learning: I—Outcome and process. *British Journal of Educational Pschology, 46*(1), 4-11. Retrieved from https://doi.org/10.1111/j.2044-8279.1976.tb02980.x

Mather, R. (2015). A mixed-methods exploration of an environment for learning computer programming. *Journal of Research in Learning Technology, 23*.

Mbunge, E., Fashoto, S., & Olaomi, J. (2021). COVID-19 and Online Learning: Factors influencing students' academic performance in first-year computer programming courses in higher education. *Available at SSRN 3757988*

Medeiros, R. P., Ramalho, G. L., & Falcão, T. P. (2018). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education, 62*(2), 77-90.

Musil, C. M., Jones, S. L., & Warner, C. D. (1998). Structural equation modeling and its relationship to multiple regression and factor analysis. *Research in Nursing & Health, 21*(3), 271-281.

Nielsen, T. (2018). The intrinsic and extrinsic motivation subscales of the Motivated Strategies for Learning Questionnaire: A Rasch-based construct validity study. *Cogent Education, 5*(1), 1504485.

Norman, G. (2010). Likert scales, levels of measurement and the "laws" of statistics. *Advances in Health-Sciences Education, 15*(5), 625-632.

Ochieng, P. A. (2009). An analysis of the strengths and limitation of qualitative and quantitative research paradigms. *Problems of Education in the 21st Century, 13* 13.

Pawlowski, J. M. (2007). The quality adaptation model: adaptation and adoption of the quality standard ISO/IEC 19796-1 for learning, education, and training. *Journal of Educational Technology & Society, 10*(2), 3-16.

Peng, J., Wang, M., & Sampson, D. (2017). Visualizing the complex process for deep learning with an authentic programming project. *Journal of Educational Technology Society 20*(4), 275-287.

Peter, J. P. (1981). Construct validity: A review of basic issues and marketing practices. *Journal of Marketing Research 18*(2), 133-145.

Rahman, M. S. (2020). The advantages and disadvantages of using qualitative and quantitative approaches and methods in language "testing and assessment" research: A literature review.

Ramalingam, V., LaBelle, D., & Wiedenbeck, S. (2004, June). Self-efficacy and mental models in learning to program. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, 171-175.

Ramalingam, V., & Wiedenbeck, S. (1998). Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy. *Journal of Educational Computing Research, 19*(4), 367-381.

Ranjeeth, S. (2011). The Impact of Learning Styles on the Acquisition of Computer Programming Proficiency. *Alternation 336*(18), 336-353.

Rego, A., Sousa, F., Marques, C., & e Cunha, M. P. (2012). Authentic leadership promoting employees' psychological capital and creativity. *Journal of Business Research 65*(3), 429-437.

Robins, A., Haden, P., & Garner, S. (2006). Problem distributions in a CS1 course. In *Proceedings of the 8th Australasian Conference on Computing Education,* 52, 165-173.

Romero, M., Lepage, A., & Lille, B. (2017). Computational thinking development through creative programming in higher education. *International Journal of Educational Technology in Higher Education, 14*(1), 1-15.

Rouder, J., Saucier, O., Kinder, R., & Jans, M. (2021). What to Do With All Those Open-Ended Responses? Data Visualization Techniques for Survey Researchers. *Survey Practice* 25699.

Ryan, R. M., & Deci, E. L. (2000). Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary Educational Psychology 25*(1), 54-67.

Saldaña, J. (2013). *The Coding Manual for Qualitative Researchers* (2nd ed.). London: Sage.

Saunders, Lewis, P., & Thornhill, A. (2009). *Research Methods for Business Students* (5th ed.). Essex, England: Pearson Education.

Saunders, M. (2011). *Research Methods for Business Students* (5th ed.). India: Pearson Education India.

Saunders, M., Lewis, P., & Thornhill, A. (2018). *Research Methods* Harlow; Munich: Pearson.

Sekaran, U., & Bougie, R. (2016). *Research Methods for Business : A Skill-building Approach* (7th ed.). UK: John Wiley & Sons.

Spada, M. M., & Moneta, G. B. (2012). A metacognitive-motivational model of surface approach to studying. *Educational Psychology, 32*(1), 45-62.

Stapleton, C. D. (1997). *Basic Concepts and Procedures of Confirmatory Factor Analysis*. Retrieved from Meeting of the Southwest Educational Research Association: Austin,Texas: http://files.eric.ed.gov/fulltext/ED407416.pdf

Streiner, D. L. (2005). Finding our way: an introduction to path analysis. *The Canadian Journal of Psychiatry, 50*(2), 115-122.

Stuckey, H. L. (2015). The second step in data analysis: Coding qualitative research data. *Journal of Social Health and Diabetes, 3*(01), 007-010.

Sullivan, G. M., & Artino Jr, A. R. (2018). Analyzing and interpreting data from Likert-type scales. *Journal of Graduate Medical Education 5*(4), 541-542.

Swanson, R. A., & Holton, E. F. (2005). *Research in Organizations: Foundations and Methods in Inquiry*. California: Berrett-Koehler Publishers.

Tavares, P. C., Henriques, P. R., & Gomes, E. F. (2017). A Computer Platform to Increase Motivation in Programming Students-PEP. In *CSEDU,* 1, 284-291.

Thu, P. T. B., Dang, L. A., Le, T. M. H., & Le, T. H. (2021). Determining Factors Affecting Teacher’s Behavioral Intention of Using Information Technology in Lecturing–Case Study of Economic Universities in Vietnam. *New Innovations in Economics, Business and Management* 25.

Tranmer, M., & Elliot, M. (2008). Multiple linear regression. *The Cathie Marsh Centre for Census and Survey Research (CCSR), 5*(5), 1-5.

Trigwell, K., Prosser, M., & Waterhouse, F. (1999). Relations between teachers' approaches to teaching and students' approaches to learning. *Higher Education, 37*(1), 57-70.

Tsai, M.-J., Wang, C.-Y., & Hsu, P.-F. (2019). Developing the computer programming self-efficacy scale for computer literacy education. *Journal of Educational Computing Research, 56*(8), 1345-1360.

Tukiainen, M., & Mönkkönen, E. (2002). Programming Aptitude Testing as a Prediction of Learning to Program. In *PPIG*, 4.

Vanthournout, G., Coertjens, L., Gijbels, D., Donche, V., & Van Petegem, P. (2013). Assessing students' development in learning approaches according to initial learning profiles: a person-oriented perspective. *Studies in Educational Evaluation, 39*(1), 33-40.

Vee, A. (2013). Understanding computer programming as a literacy. *Literacy in Composition Studies, 1*(2), 42-64. Retrieved from http://d-scholarship.pitt.edu/21695/1/24-33-1-PB.pdf

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical Engineering Sciences, 366*(1881), 3717-3725. Retrieved from https://doi.org/10.1098/rsta.2008.0118

Yacob, A., & Saman, M. Y. M. (2012). Assessing level of motivation in learning programming among engineering students. In *The International Conference on Informatics and Applications (ICIA2012)*, 425-432. The Society of Digital Information and Wireless Communication.

Youngman, M. (1979). A Comparison of Item-Total Point Biserial Correlation, Rasch and Alpha-Beater Item Analysis Procedures. *Educational Studies, 5*(3), 265-273.

Zahedi, L., Batten, J., Ross, M., Potvin, G., Damas, S., Clarke, P., & Davis, D. (2021). Gamification in education: a mixed-methods study of gender on computer science students' academic performance and identity development. *Journal of Computing in Higher Education* 1-34. Retrieved from https://doi.org/10.1002/9781119171386.ch19

APPENDIX A – Survey Questionnaire

SECTION A: Demographic & Background Information

Please select the option that would best fit your response by making a tick (✓) in the box provided and fill in required information

Part 1:

College of Affiliation	Law & Management			
	Humanities			
	Agriculture, Engineering and Science			
Degree	B. Com	B. Sc	Hons	MCom
Campus	Westville		PMB	
Gender	Male		Female	
Age	19-21	21-23	23-26	

Part 2:

*How many **years of experience** do you have in using the following computing tools/techniques (Please tick (✓) the relevant cell)*

	0-2 years	*2-3 years*	*3-4 years*	*More than 4 years*
Computer Programming				
Database Design				
Systems Analysis and Design				

Part 3:

Level of Achievement

	90-100	*80-89*	*70-79*	*60-69*	*50-59*	*40-49*	*40-49*	*Below 49*
On an average, what is your approximated level (%) of ***achievement in computer programming***								
On an average, what is your approximated level (%) of ***achievement in systems analysis and design***								

SECTION B: Behavioural, Cognitive Factors and Problem Solving

Part 1: Intrinsic and Extrinsic motivation

Please select the option that would best fit your motivation style by making a tick (✓) in the box provided.

Indicate your preference when it comes to ***learning computer programming and understanding of technology***	***Strongly Agree***	***Agree***	***Neutral***	***Disagree***	***Strongly Disagree***
I prefer course material that really challenges me so I can learn new things					
When I don't understand something right away I try to figure it out by myself					
I prefer course material that arouses my curiosity even if it is difficult to learn					
Getting good marks for programming brings me a sense of personal satisfaction					
I want to do well in my programming modules because it is important to show my ability to my family, friends and lecturers					
I engage with new technology because that is what society expects of me					
I engage with new technology so that I have a sense of control over the technology					
I make an effort to master computer programming so that I can fit in with other students in my class					

Part 2: Learning style

Please select the option that would best fit your motivation style by making a tick (✓) in the box provided.

*Indicate your preferences when it comes to the **studying of course content***	***Strongly Agree***	***Agree***	***Neutral***	***Disagree***	***Strongly Disagree***
I find most new topics interesting and will often spend extra time trying to understand how they work					
I find it helpful to study topics in depth rather than trying to remember important facts for tests					
I test myself on important topics until I understand them completely					
I tend to study best by using memorisation techniques					
I find the best way to pass tests is trying to learn the answers to likely questions					
I prefer to ensure that I pass a course even though my understanding of concepts may not be very good					

Part 3: Self-Efficacy

Please select the option that would best fit your level of intrinsic and extrinsic motivation by making a tick (✓) in the box provided.

*Indicate your response to the questions **on self-efficacy** and computer programming*	***Strongly Agree***	***Agree***	***Neutral***	***Disagree***	***Strongly Disagree***
I am confident of my ability to develop suitable strategy for a given programming task in a short time					
I am able to construct programming code that is logically correct					
I have the capacity to easily identify errors in my programming code.					
I am able to mentally trace through the execution of a long, complex program					
I could organize and design my program in a modular/procedural manner					
I have a good understanding of the object-oriented paradigm for programming					
I could rewrite lengthy and confusing portions of code to be more readable and clearer.					
I am confident of my ability to identify the objects in the problem domain and declare, define, and use them.					
I am able to write computer programming code to sort out a given set of numbers into ascending/descending order					
I feel that I am better at programming when I get the help of someone else.					
I feel more comfortable to complete a programming problem if someone showed me how to solve the problem first.					

I could manage my time efficiently if I had a pressing deadline on a programming project					

Part 4: Problem Solving and Mental Model

1. Assume that **i** is an integer

Identify the correct ***list of integers*** that meet the following conditions:

(i> =1) **AND** (i< 5)

Choose from the following options	Select the correct answer with a ✓
Option 1: 1; 2; 3; 4; 5	
Option 2: 1; 2; 3; 4;	
Option 3: 2; 3; 4;	
Option 4: This condition is impossible to satisfy	
Option 5: An infinite set of numbers	

2. Assume that **j** is an integer

Identify the correct ***list of integers*** that meet the following conditions:

(j>=3) **OR** (j<7)

Choose from the following options	Select the correct answer with a ✓
Option 1: 3; 4; 5; 6; 7	
Option 2: 3; 4; 5; 6	
Option 3: Integers greater than and equal to 3 AND less than 7	
Option 4: Any integer would meet satisfy this condition	
Option 5: This condition is impossible to satisfy	

3. Examine the following series of characters and ***predict the NEXT item*** in the series

bce, bbcde, bbbcdde,……….

Choose from the following options	Select the correct answer with a ✓
Option 1: bbbbcddde	
Option 2: bbbccddee	
Option 3: bbbccddde	
Option 4: There is not enough information to predict the next item	
Option 5: There could be more than one item that may be regarded as the next item	

4. Examine the following series of characters and ***predict the NEXT item*** in the series

bcace, bcacacace, bcacacacacace,…….

Choose from the following options	Select the correct answer with a ✓
Option 1: bcacecacecaceca	
Option 2: bcacacacacacacace	
Option 3: bcacecacacacece	
Option 4: There is not enough information to predict the next item	
Option 5: There could be more than one item that may be regarded as the next item	

5. Examine the following series of characters and ***predict the FIRST item*** in the series

………, abcccdd , abbccccdd , abbbcccccdd

Choose from the following options	Select the correct answer with a ✓
Option 1: bccdd	
Option 2: abccd	
Option 3: abbccdd	
Option 4: accdd	
Option 5: There could be more than one item that may be regarded as the first item	

6. How many ***comparisons*** would it take to ***determine the smallest number*** in each of the following situations?

a) There are ***3 numbers in a list of numbers***

1	2	3	4	5

b) There are ***5 numbers in a list of numbers***

4	5	6	7	More than 7

b) There are ***n numbers in a list of numbers*** (n>=1)

1	n-1	n	n+1	More than n+1

7. You are required to ***compute the sum of 20 numbers***.
Which of the options listed below is the best strategy to follow:

Choose from the following options	Select the correct answer with a ✓
Option 1: You would need to use an array of 20 numbers and sum all the elements in the array	
Option 2: You could use a loop that executes 20 times and a variable that represents the sum and add all the numbers to this variable	
Option 3: You could use 20 individual variables and add all of these variables together to give you the sum	
Option 4: You could use a single variable and 20 input statements and add all the values that are input into the single variable	
Option 5: You could use an if then statement and as soon as the condition that 20 numbers have been added, you could stop execution	

8. You are required to ***compute the average*** of a set of numbers that are input via the keyboard; the input will stop as soon as the number zero is input. ***Which looping structure will be an ideal choice?*** You may choose from:

Choose from the following options	Select the correct answer with a ✓
For Loop	
While Lop	
Foreach loop	
Nested Loop	

9. Assume that 2 variables are declared and given values as follows:

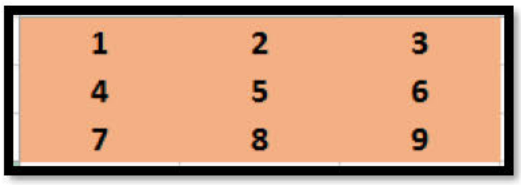
int x=10; int y=20

You would like to write programming code so that the ***values are switched*** i.e. x becomes 20 and y becomes 10. Which of the following options is the correct way of doing this?

Option 1: x=y; y=x;	
Option 2: int temp=x; x=y; y=temp;	
Option 3: int.Swap (x,y)	
Option 4: int temp=x; y=x; x=y;	

10. Assume that the following values have been stored in a 2D data structure named ***TwoD***

TwoD



1	2	3
4	5	6
7	8	9

Determine the output that will be produced by the following programming segment

```
int sum=0;
for (int x = 0; x < 3; x++)
    sum += TwoD[x, x];
Console.WriteLine(sum);
```

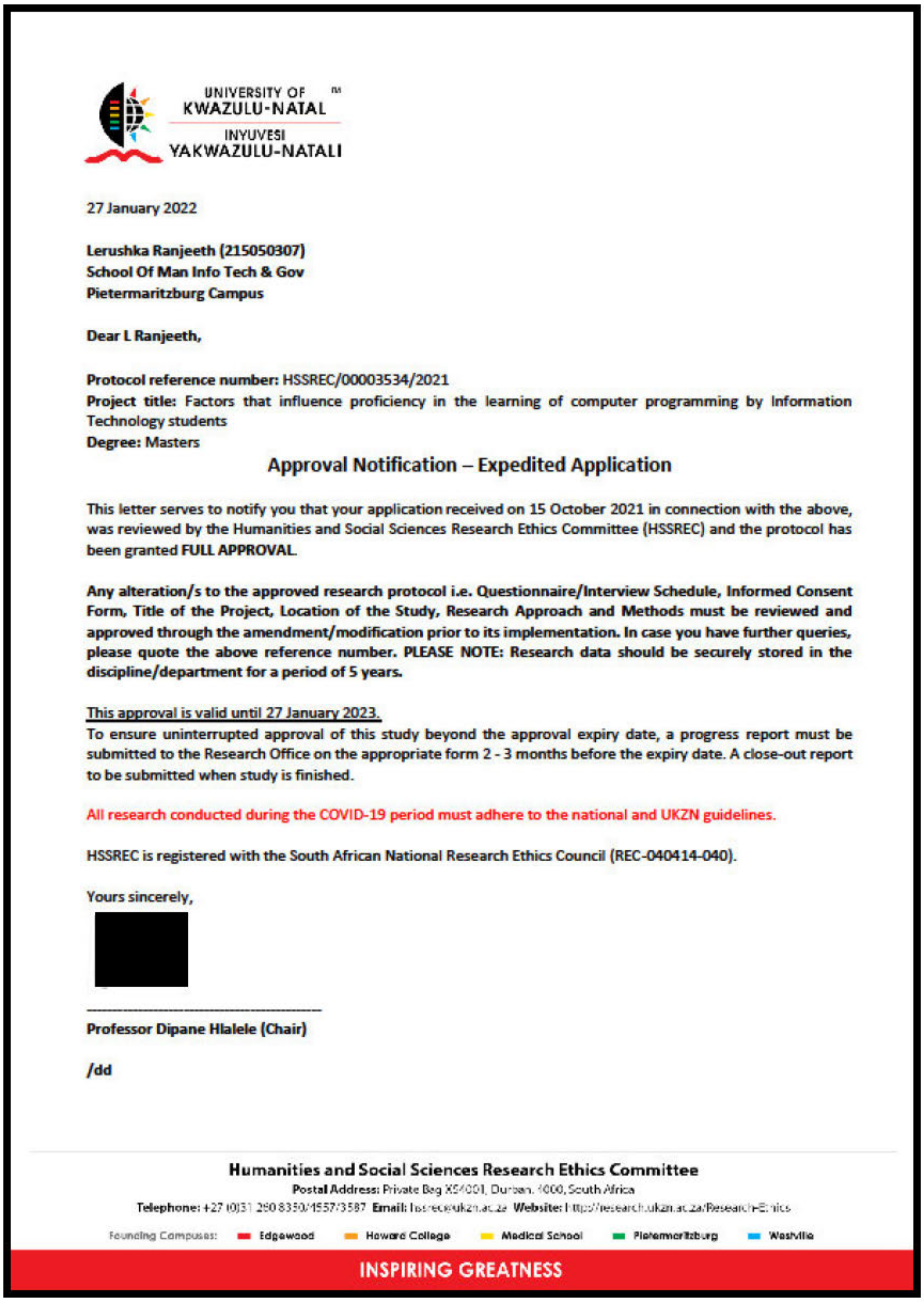
Choose from the following options	Select the correct answer with a ✓
Option 1: ***9***	
Option 2: **15**	
Option 3: ***0***	
Option 4: ***3***	
Option 5: **An error will be output**	

SECTION C: Open Ended Response

Please provide any suggestions that you would like to make to help better understand your approach/challenges with regards to learning/mastery of computer programming

Thank You for your Responses!

APPENDIX B – Ethical Clearance



UNIVERSITY OF KWAZULU-NATAL
INYUVESI YAKWAZULU-NATALI

27 January 2022

Lerushka Ranjeeth (215050307)
School Of Man Info Tech & Gov
Pietermaritzburg Campus

Dear L Ranjeeth,

Protocol reference number: HSSREC/00003534/2021
Project title: Factors that influence proficiency in the learning of computer programming by Information Technology students
Degree: Masters

Approval Notification – Expedited Application

This letter serves to notify you that your application received on 15 October 2021 in connection with the above, was reviewed by the Humanities and Social Sciences Research Ethics Committee (HSSREC) and the protocol has been granted **FULL APPROVAL**.

Any alteration/s to the approved research protocol i.e. Questionnaire/Interview Schedule, Informed Consent Form, Title of the Project, Location of the Study, Research Approach and Methods must be reviewed and approved through the amendment/modification prior to its implementation. In case you have further queries, please quote the above reference number. PLEASE NOTE: Research data should be securely stored in the discipline/department for a period of 5 years.

This approval is valid until 27 January 2023.
To ensure uninterrupted approval of this study beyond the approval expiry date, a progress report must be submitted to the Research Office on the appropriate form 2 - 3 months before the expiry date. A close-out report to be submitted when study is finished.

All research conducted during the COVID-19 period must adhere to the national and UKZN guidelines.

HSSREC is registered with the South African National Research Ethics Council (REC-040414-040).

Yours sincerely,

[redacted]

Professor Dipane Hlalele (Chair)

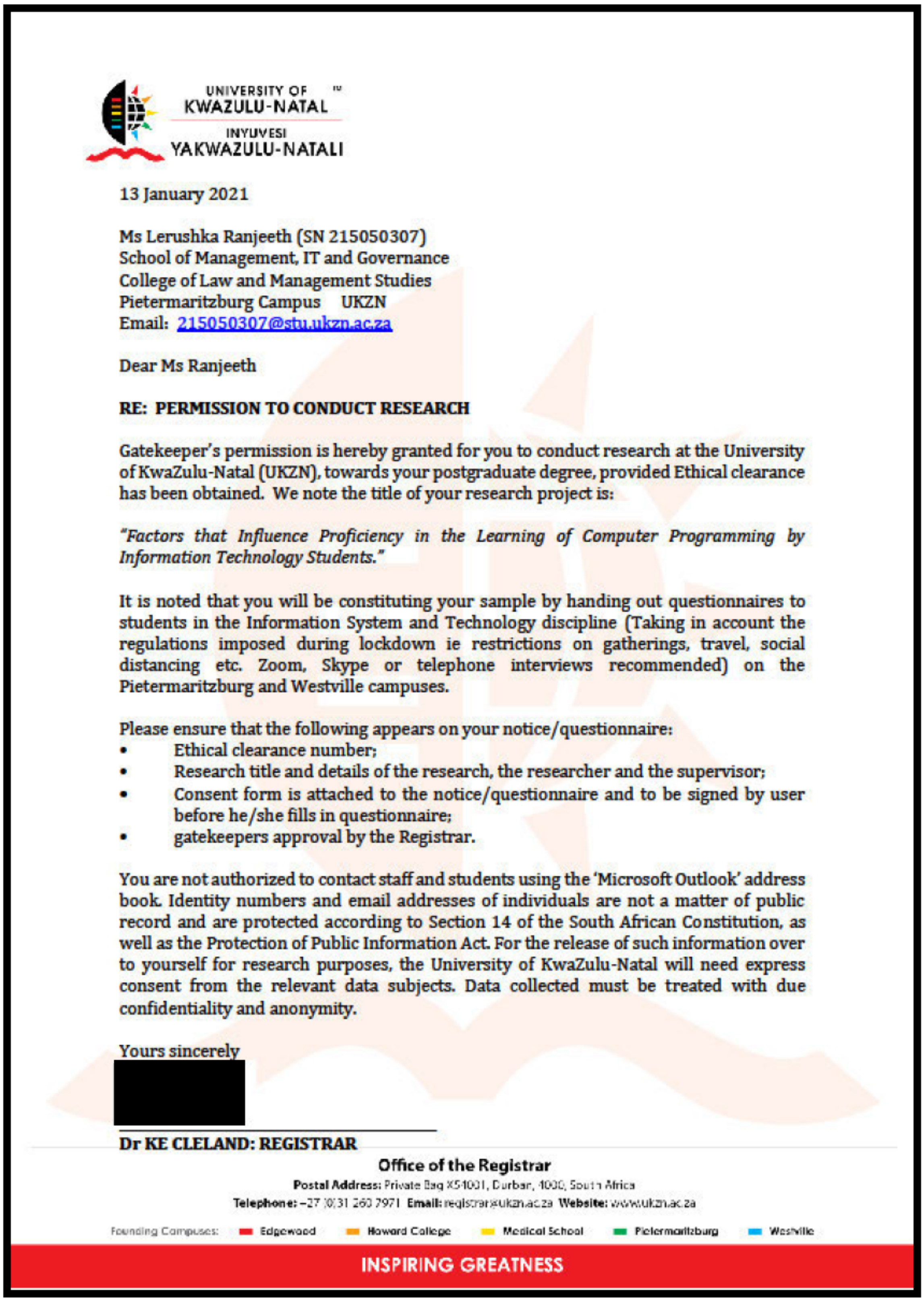
/dd

Humanities and Social Sciences Research Ethics Committee
Postal Address: Private Bag X54001, Durban, 4000, South Africa
Telephone: +27 (0)31 260 8350/4557/3587 **Email:** hssrec@ukzn.ac.za **Website:** http://research.ukzn.ac.za/Research-Ethics

Founding Campuses: 🟥 Edgewood 🟧 Howard College 🟨 Medical School 🟩 Pietermaritzburg 🟦 Westville

INSPIRING GREATNESS

APPENDIX C – Gate Keeper Clearance



UNIVERSITY OF KWAZULU-NATAL
INYUVESI YAKWAZULU-NATALI

13 January 2021

Ms Lerushka Ranjeeth (SN 215050307)
School of Management, IT and Governance
College of Law and Management Studies
Pietermaritzburg Campus UKZN
Email: 215050307@stu.ukzn.ac.za

Dear Ms Ranjeeth

RE: PERMISSION TO CONDUCT RESEARCH

Gatekeeper's permission is hereby granted for you to conduct research at the University of KwaZulu-Natal (UKZN), towards your postgraduate degree, provided Ethical clearance has been obtained. We note the title of your research project is:

"Factors that Influence Proficiency in the Learning of Computer Programming by Information Technology Students."

It is noted that you will be constituting your sample by handing out questionnaires to students in the Information System and Technology discipline (Taking in account the regulations imposed during lockdown ie restrictions on gatherings, travel, social distancing etc. Zoom, Skype or telephone interviews recommended) on the Pietermaritzburg and Westville campuses.

Please ensure that the following appears on your notice/questionnaire:

- Ethical clearance number;
- Research title and details of the research, the researcher and the supervisor;
- Consent form is attached to the notice/questionnaire and to be signed by user before he/she fills in questionnaire;
- gatekeepers approval by the Registrar.

You are not authorized to contact staff and students using the 'Microsoft Outlook' address book. Identity numbers and email addresses of individuals are not a matter of public record and are protected according to Section 14 of the South African Constitution, as well as the Protection of Public Information Act. For the release of such information over to yourself for research purposes, the University of KwaZulu-Natal will need express consent from the relevant data subjects. Data collected must be treated with due confidentiality and anonymity.

Yours sincerely

[redacted]

Dr KE CLELAND: REGISTRAR

Office of the Registrar
Postal Address: Private Bag X54001, Durban, 4000, South Africa
Telephone: +27 (0)31 260 7971 **Email:** registrar@ukzn.ac.za **Website:** www.ukzn.ac.za

Founding Campuses: 🟥 Edgewood 🟧 Howard College 🟨 Medical School 🟩 Pietermaritzburg 🟦 Westville

INSPIRING GREATNESS